J. Symbolic Computation (1999) **28**, 401–433
Article No. jsco.1999.0289
Available online at http://www.idealibrary.com on IDEAL

Application of Computer Algebra Systems for Stability Analysis of Difference Schemes on Curvilinear Grids

VICTOR G. GANZHA[§¶†] AND EVGENII V. VOROZHTSOV[‖‡]

[†] *Institute of Informatics, Technical University of Munich, Munich, 80290 Arcisstraße 21, Germany*
[‡] *Institute of Theoretical and Applied Mechanics, Russian Academy of Sciences, Novosibirsk, 630090, Russia*

The paper deals with problems arising in the application of the computer algebra systems for the symbolic–numeric stability analysis of difference schemes and schemes of the finite-volume method approximating the two-dimensional Euler equations for compressible fluid flows on curvilinear spatial grids. We carry out a detailed comparison of the REDUCE 3.6 and *Mathematica* (Versions 2.2 and 3.0) from the point of view of their applicability to the solution of the above problems. We draw a conclusion that a preference should be given for *Mathematica* from the viewpoint of the execution of symbolic–numeric computations. We also describe in detail our new symbolic–numeric algorithm for stability investigation, which was implemented with the aid of *Mathematica*. The proposed method enables us to reduce the needed computer storage at the symbolic stages by a factor of about 20 as compared with the previous algorithms. A feature of the numerical stages is the use of the arithmetic of rational numbers, which enables us to avoid the accumulation of the roundoff errors. We present the examples of the application of the proposed symbolic–numeric method for stability analysis of very complex schemes of the finite-volume method on curvilinear grids, which are widely used in computational fluid dynamics.

1. Introduction

The development of accurate and efficient methods for numerical solution of the transonic aerodynamics problems is one of the difficult problems of computational fluid dynamics. A feature of these problems is that curved boundaries are present in the spatial integration region. The consideration of these boundaries can be performed in the most efficient way with the aid of the curvilinear spatial computational grids. The curvilinear grids consisting of the quadrilateral cells have gained a very wide acceptance. They are mapped with the aid of a non-degenerate transformation

$$x = x(\xi, \eta), \qquad y = y(\xi, \eta) \tag{1}$$

[§] Author to whom correspondence should be addressed.
[¶] E-mail: ganzha@informatik.tu-muenchen.de
[‖] E-mail: vorozh@itam.nsc.ru

0747–7171/99/090401 + 33 $30.00/0

from the plane of the Cartesian spatial coordinates (x, y) onto a uniform rectangular grid in the plane of curvilinear coordinates ξ, η. Such curvilinear grids are also called the logically rectangular grids (Knupp and Steinberg, 1994).

One of the basic ways for increasing computational efficiency of numerical methods on logically rectangular grids is to increase the robustness of stability. Then the stationary solution of the Euler equations of inviscid fluid can be achieved faster with the aid of the pseudo-unsteady method at the expense of using larger time steps (Fletcher, 1996).

The stability interval of a finite-difference method for the numerical integration of the Euler equations can be increased by using fully implicit schemes. However, in the case of the hyperbolic systems of partial differential equations the requirement of diagonal dominance of the matrices arising in the numerical algorithms may impose limitations on the time steps (Strikwerda, 1989) or may be violated in the splitting-up schemes (Kovenya and Yanenko, 1981).

The explicit difference schemes are much easier to implement, but the stability condition imposes restrictions on the time steps. A goal-oriented design of the explicit highly accurate schemes possessing a high stability robustness for multidimensional problems solved on curvilinear grids involves an extraordinarily complex problem of the stability investigation of such schemes. This is related to the fact that the difference approximations of the Euler equations on curvilinear grids have a much more complex form than in the case of the uniform rectangular grids. This leads to a dramatic increase of the complexity of algebraic expressions arising in the process of the stability investigation of difference schemes on curvilinear grids with the aid of the Fourier method.

As a result of this, until now only a few comparatively simple difference schemes on curvilinear grids have been analysed; a review of these studies may be found in Ganzha and Vorozhtsov (1996a). Further progress in the development of highly stable explicit methods of computational fluid dynamics will depend on whether it will be possible to find an efficient solution of the problem of stability investigation of complex difference schemes.

Our experience summarized in Ganzha and Vorozhtsov (1996a) shows that computer algebra systems are indeed the only means which enables one to obtain data on stability even for difference schemes for three-dimensional problems on curvilinear grids. We have described in the above-mentioned book the implementation of various symbolic–numeric methods for stability investigation with the aid of REDUCE and FORTRAN.

In 1988, a new computer algebra system (CAS) *Mathematica* appeared. This drew our attention because it possesses the capabilities for the numerical computations in the machine arithmetic of floating-point numbers with any accuracy. In this connection we have made a number of successful attempts at the implementation of some of the symbolic–numeric methods presented in Ganzha and Vorozhtsov (1996a) with the aid of *Mathematica* 2.2 (Wolfram, 1991). However, in the process of implementing the above methods within the framework of the *Mathematica* system we have faced certain shortcomings of this system.

In the present paper we describe both the advantages and shortcomings of the system *Mathematica* and propose ways to circumvent the arising problems. We also compare REDUCE and *Mathematica* from the viewpoint of their general applicability for implementation within the framework of a single CAS of all the stages of the symbolic–numeric algorithms for the stability analysis of difference schemes for solving the multidimensional problems of computational fluid dynamics on curvilinear logically rectangular spatial computing meshes.

This paper is organized as follows. In Section 2, we discuss the peculiarities of the numerical solution of aerodynamical problems on curvilinear spatial grids. In Section 3 we construct with the aid of *Mathematica* all known types of curvilinear grids around an isolated airfoil. The issues related to the application of the Fourier stability analysis for difference schemes on curvilinear grids are addressed in Section 4. A Jameson explicit three-stage finite-volume scheme for the numerical integration of two-dimensional Euler equations is presented in Section 5. We also perform a linearization of arising difference equations and investigate their stability by Fourier method. In Section 6, we present a new symbolic–numeric stability analysis method for the cases of finite-difference or finite-volume schemes on arbitrary curvilinear grids of quadrilateral cells. We discuss specific implementation issues related both to symbolic and numerical stages of our method. In Section 7 we present the results of the application of the presented symbolic–numeric method to stability analysis of a Jameson scheme for the problem of gas flow around an airfoil. In Section 8, we compare *Mathematica* and REDUCE. In Section 9, we summarize the results obtained and give an overview of extensions and further research.

2. Numerical Solution of Aerodynamical Problems

The governing equations of fluid flows are non-linear partial differential equations. The analytical solution of these equations is possible only for very particular cases of flow problems. This is the main reason why the numerical methods for the solution of the governing equations of fluid flows are the basic means for obtaining solutions of practically important aerodynamics problems.

The process of the development of a computer program for the numerical solution of a specific aerodynamics problem on a curvilinear grid consists of the following three main stages:

(1) The development of a program for the numerical generation of a curvilinear spatial grid;
(2) The development of a program implementing a chosen numerical method for the solution of the Euler or Navier–Stokes equations on a given grid;
(3) The development of a program for the graphical visualization of the computational results, the computation of the needed integral functionals (the lift force coefficients, the drag coefficients, etc.).

The problem on the development of efficient numerical algorithms for the curvilinear grid generation is in itself sufficiently complex (Thompson *et al.*, 1985; Knupp and Steinberg, 1994). We note here two general peculiarities of the numerical generation of curvilinear grids: (i) the explicit analytic form of the functions (1) is specified in the computer codes for numerical grid generation only in very rare cases; the algorithms for grid generation enable one as a rule to obtain only the numerical values of the (x, y) coordinates in the grid nodes, that is at points of the intersection of the coordinate lines $\xi = \text{const}$ and $\eta = \text{const}$; (ii) the coordinates (x_{jk}, y_{jk}) of the grid nodes are the machine floating-point numbers; here and in what follows j and k are the numbers of the grid node in the direction of the ξ and η axes, respectively. As we will show below, these coordinates are used directly at the computation of the local time steps of the difference scheme, which are the maximum values allowed by stability.

Since the numerical grid generation is needed also for the implementation of a symbolic–numeric stability analysis of difference schemes on curvilinear grids it is natural to require that the algorithms for grid generation are also implemented within the framework of the same CAS, which is to be used for the stability analysis of a difference scheme on curvilinear grid. The CAS *Mathematica 2.2* proved to be a very appropriate tool for rapid generation of curvilinear grids.

At the stage of the choice of a numerical method for the solution of the Euler or Navier–Stokes equations, one can choose between three basic classes of numerical methods:

finite-difference methods;
finite-volume methods;
finite-element methods.

Within each of these classes, there are the methods possessing different orders of accuracy with respect to a reference size of the cells of a spatial computing mesh. The higher-order (second-order, third-order, etc.) methods produce excellent results in the case of smooth solutions.

However, strong discontinuities (shock waves, contact discontinuities) can arise in many fluid dynamics problems even at smooth initial conditions (Richtmyer and Morton, 1967; Roache, 1976; LeVeque, 1992). In these cases, the parasitic or spurious oscillations arise in the numerical solutions near the strong discontinuities. These oscillations are caused by the dispersion (Strikwerda, 1989; Ganzha and Vorozhtsov, 1996b) and Gibbs phenomenon (Vorozhtsov and Yanenko, 1990). In order to suppress the spurious oscillations one usually introduces in the schemes of the finite-difference, finite-volume or finite-element method the extra terms representing the artificial viscosity or dissipation. These terms can have a different effect on the stability properties of a specific numerical method depending on the structure of both the discretization scheme of the method, the form of the artificial dissipator terms, and the strength of shock waves arising in fluid flow (Richtmyer and Morton, 1967; Ganzha *et al.*, 1994). The introduction of artificial (and/or physical) dissipation terms can lead either to a reduction or increase of the size of the stability region.

The consideration of artificial dissipation terms in the stability analyses of numerical methods leads to a significant increase in the complexity of these analyses. However, our experience shows that the consideration of artificial dissipators within the framework of the Fourier method applied to the linearized difference equations does not lead to new problems from the viewpoint of computer algebra: only the length of algebraic expressions appearing in computer implementation of the Fourier method increases. This is a purely technical problem, which can be solved successfully with the aid of the same techniques, which are presented below in Section 6 of this paper.

Another reason why we consider the Jameson's finite-volume schemes in the absence of artificial dissipation terms is as follows. It was stressed by Pike and Roe (1985) that the Jameson's schemes belong to those rare schemes, for which it is possible to obtain the necessary von Neumann stability condition in closed form. This is extremely important for the purpose of the validation of new symbolic–numerical methods for stability analysis of numerical methods. The optimal structure of Jameson's schemes in the absence of artificial dissipation has enabled us to obtain the necessary stability condition (35) of these schemes on a curvilinear grid under certain sufficiently reasonable limitation on the grid topology (namely, the curvilinear grid should be a parallelogram grid).

Therefore, we concentrate below on the Jameson's finite-volume schemes without artifi-

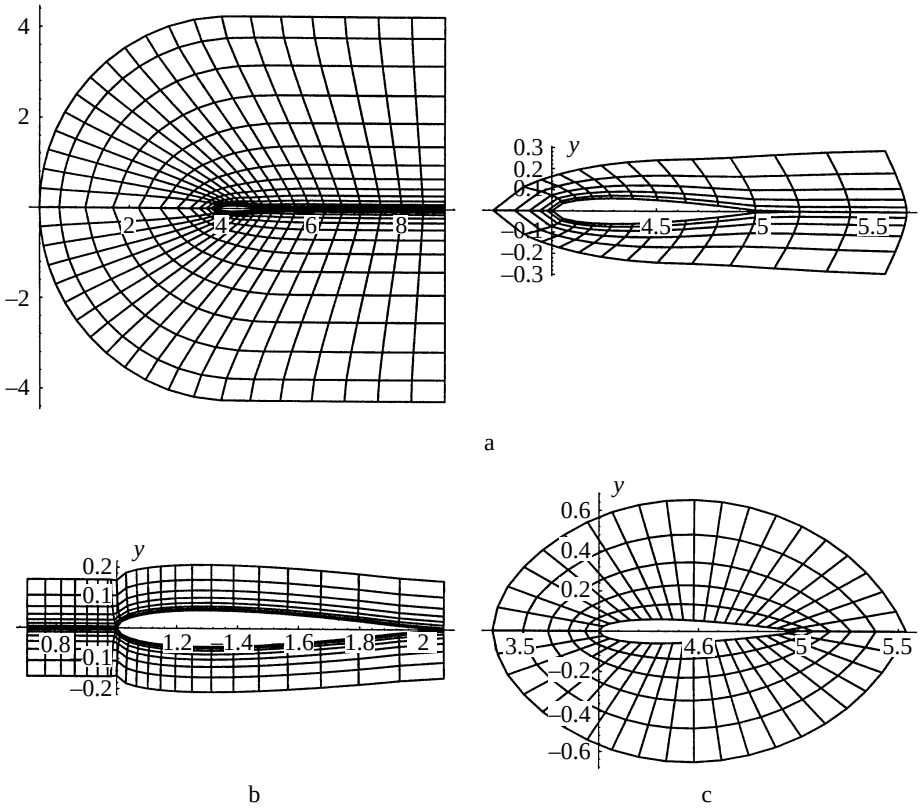

Figure 1. The grids around the airfoil NACA0012: a, C-grid of 41 × 15 nodes; b, H-grid; c, O-grid.

cial dissipators and we will describe at the example of these schemes the typical problems arising in stability analyses with the aid of computer algebra.

3. Curvilinear Grid and Topologies

In this paper we consider the application of symbolic–numeric algorithms for the stability investigation of difference methods for the numerical solution of a problem on the transonic fluid flow around an airfoil. The choice of this problem is related to the fact that the vast majority of current numerical solvers for this problem use either curvilinear logically rectangular grids (Thompson *et al.*, 1985) or unstructured grids (Mavriplis, 1988) (i.e. grids consisting of the triangular cells of different sizes).

For problems regarding the fluid flow around an isolated airfoil the following three topologically different types of grids are used: the C-grids, H-grids, and O-grids. In Figure 1 we show all these types of grids around the airfoil NACA0012 (the function specifying the geometry of this profile may be found in Fletcher, 1996). In the cases of the H- and O-grids we show in Figure 1 for brevity sake only the partial view of the grids near the airfoil.

The C-grid (Figure 1a) was generated with the aid of the multisurface method (Eiseman, 1979; Fletcher, 1996). On the vertical boundary segment downstream from the

airfoil the grid nodes were computed with the use of the effective stretching function

$$s = P_{\eta^*} + (1 - P)\left(1 - \frac{\tanh[Q(1 - \eta^*)]}{\tanh Q}\right), \tag{1}$$

where $0 \leq \eta^* \leq 1$ and the user-specified parameters P and Q provide grid point control. In our *Mathematica* program the values s_i of the stretching function (1) are computed as machine floating-point numbers with the aid of the following function:

```
Stretch[np_,p_,q_]:=( s={}; an=np-1; deta=1./an; tqi=N[1./Tanh[q]];
Do[ al=i-1; eta=al deta; dum=q (1-eta); dum=N[1-Tanh[dum] tqi];
si=N[p eta + (1-p) dum]; AppendTo[s,si],{i,np}]; s );
```

The curvilinear grid lines $\eta = \text{const}$ wrap around the airfoil, whereas the lines $\xi = \text{const}$ emanate from the airfoil surface and from the cut line behind the trailing edge of the airfoil.

The H-topology mesh of Figure 1b was also generated with the aid of the multisurface method (Eiseman, 1979; Fletcher, 1996). The same stretching function (1) was used on vertical boundary segment downstream from the airfoil.

The O-topology mesh of Figure 1c was generated with the aid of linear transfinite interpolation (Thompson *et al.*, 1985; Knupp and Steinberg, 1994). The grid points on the horizontal cut line behind the trailing edge of the airfoil were specified with the aid of the stretching function (1). We now present the program fragment, which implements the method of transfinite interpolation.

```
Do[ a1=1-xx[[j,1]]+R;  Do[  b1 = saf[[k]]; xbt=xx[[j,1]]; xtt =
xx[[j,kmax]]; ybt = yy[[j,1]]; ytt = yy[[j,kmax]];
xx[[j,k]]=(1-b1)*xbt+b1*xtt+(1-a1)*xx[[1,k]]+a1*xx[[jb1,k]]-(a1*b1*xx
[[jb1,kmax]]+ a1*(1-b1)*xx[[jb1,1]]+b1*(1-a1)*xx[[1,kmax]]+(1-a1)*
(1-b1)*xx[[1,1]]);yy[[j,k]]=(1-b1)*ybt+b1*ytt+(1-a1)*yy[[1,k]]+a1*
yy[[jb1,k]]-(a1*b1*yy[[jb1,kmax]]+a1*(1-b1)*yy[[jb1,1]]+b1*(1-a1)*
yy[[1,kmax]]+(1-a1)*(1-b1)*yy[[1,1]]),{k,2,kma1}],{j,jb1}];
```

It is important that obtaining the coordinates `(xx[[j,k]], yy[[j,k]])` of grid nodes in the form of floating-point machine numbers presents no difficulties when using the *Mathematica* system, whereas in the case of the REDUCE system this would be very problematic.

Since the above three grid topologies differ significantly from each other, there is a problem in the comparison of stability analysis results on these three grids. In order to achieve the maximum "resemblance" of all the three grids we satisfy the following conditions during generation of the grids of Figure 1:

(a) in all grids, the distance along the x- and y-axes from the airfoil surface to the external boundary is the same and is equal to four chord lengths of the airfoil;

(b) the same number of the grid nodes is used in the case of C- and O-grids; in the case of H-grid, the total number of grid nodes located on the nearly vertical grid lines emanating from the airfoil surface and from the horizontal cut line behind the

Table 1. The values $h_{\min}$ and h_{mean} for the types of grids considered.

Type of grid	J	K	$h_{\min}$	h_{mean}
C-grid	41	15	0.042 1328	0.327 604
H-grid	70	15	0.041 8564	0.274 946
O-grid	41	15	0.054 6367	0.283 659

airfoil trailing edge is the same as the overall number of grid points in the case of the C- and O-meshes of Figures 1a,c;

(c) the same stretching function (1) is used in all three grids as indicated above, with the parameters $P = 0.1$ and $Q = 2.0$.

Let J be the number of the grid nodes along the ξ-axis, and K the number of the grid nodes along the η-axis. Further let l_{jk} be the arithmetic mean of the lengths of sides of a quadrilateral grid cell whose vertices are (j,k), $(j+1,k)$, $(j,k+1)$ and $(j+1,k+1)$. We now introduce the following two quantitative measures of curvilinear grid density:

$$h_{\min} = \min_{j,k} l_{jk}, \qquad h_{\mathrm{mean}} = \frac{1}{(J-1)(K-1)} \sum_{j=1}^{J-1} \sum_{k=1}^{K-1} l_{jk},$$

that is h_{mean} is the mean linear size of cells of a given curvilinear grid.

In Table 1 we present the values of $h_{\min}$ and h_{mean}, which we have obtained for all three grid topologies under the above conditions (a)–(c). Both $h_{\min}$ and h_{mean} are of the same order for all three grid types.

4. Fourier Symbol on Curvilinear Grids

In our book (Ganzha and Vorozhtsov, 1996a) we have presented, analysed, and compared in detail 10 different methods of stability investigation for difference schemes approximating the partial differential equations of the mathematical physics. As a result of this analysis, we concluded that the Fourier method (Lax and Nirenberg, 1966; Thomée, 1969; Gustafsson *et al.*, 1972; Roache, 1976; Godunov and Ryabenkii, 1987; Strikwerda, 1989; Fletcher, 1996; Ganzha and Vorozhtsov, 1996a,b) is the most reliable and universal method for stability investigation of difference schemes. We will briefly present in this section the general procedure of the Fourier method and its peculiarities related to the difference schemes on curvilinear grids.

In the case of difference schemes on curvilinear grids the difference equation coefficients also contain the coordinates of the curvilinear grid nodes. If we "freeze" the components of the solution vector $\vec{u}$ in the difference scheme coefficients then we obtain the linearized difference scheme

$$\vec{u}^{n+1} = S(x, y, \vec{u}_0)\vec{u}^n, \tag{1}$$

where $\vec{u}_0$ is the "frozen" solution vector (i.e. all the components of the solution vector are assumed to be equal to certain constant values), n is the number of the time level, that is $\vec{u}^n$ is the solution computed at $t = t^n = \tau_1 + \tau_2 + \cdots + \tau_n$, $n = 1, 2, \ldots$; τ_m is the value of the time step with which the solution is advanced from time level $m-1$ to time level $m, m = 1, 2, \ldots$. The difference operator $S(x, y, \vec{u}_0)$ in (1) is called the step operator, and its structure determines to a large extent the stability properties of a difference method.

In what follows we investigate the stability of the difference initial-value problem

$$\vec{u}^{n+1} = S(x, y, \vec{u}_0)\vec{u}^n, \qquad n = 0, 1, 2, \ldots;$$
$$\vec{u}^0 = \vec{U}_0(x, y), \qquad -\infty < x, y < \infty, \tag{2}$$

where $\vec{U}_0(x, y)$ is the given initial condition at $t = 0$. Thus, we do not take into account the difference boundary conditions. There are a number of reasons, which determine the usefulness of the results obtained from the stability analysis of the difference problem (2) (see also Ganzha and Vorozhtsov, 1996a).

(i) There exists the Babenko–Gel'fand criterion (Godunov and Ryabenkii, 1987) for the stability of difference initial- and boundary-value problems. In accordance with this criterion, the stability of a corresponding difference initial-value problem is necessary for the stability of a difference initial- and boundary-value problem (see also Gustafsson *et al.*, 1972).

(ii) The additional restrictions on stability, which may be imposed by consideration of auxiliary one-sided difference initial- and boundary-value problems in the GKS analysis of (Gustafsson *et al.*, 1972) can be removed by using implicit difference formulas at the boundaries (Chakravarthy, 1983).

Since the coefficients of scheme (2) depend on x, y, the Fourier transform cannot be applied directly to this scheme. Instead, one considers the Fourier symbol of the operator S, which is obtained by fixing the values of x and y in S and by substituting a solution of the form

$$\vec{u}(x(\xi, \eta), y(\xi, \eta), t) = \vec{u}_0 \exp\{i(k_1\xi + k_2\eta - \omega t)\}$$

into (2), where the functions $x(\xi, \eta)$ and $y(\xi, \eta)$ enter the transformation (1), and k_1, k_2 are the real components of the wave vector, ω is the wave frequency, and $i = \sqrt{-1}$. In this way we find $G(\xi, \eta, \vec{u}_0)$, the Fourier symbol of the operator S; in the case of constant coefficients in S, the operator G is called the amplification matrix of the linearized difference scheme (1). The uniform boundedness of $\|G\|$ is necessary (Lax and Nirenberg, 1966; Thomée, 1969) for the stability of scheme (1). For the uniform boundedness of $\|G\|$ it is in turn necessary that the conditions

$$|\lambda_\alpha| \leq 1 + O(\tau), \qquad \alpha = 1, \ldots, M_u, \tag{3}$$

are satisfied (Godunov and Ryabenkii, 1987; Strikwerda, 1989), where λ_α are the eigenvalues of the amplification matrix G and M_u is the dimension of matrix operator G. Inequalities (3) represent the necessary von Neumann stability conditions. They should be checked locally at each point (x_{jk}, y_{jk}) on the curvilinear spatial grid.

5. The Three-stage Jameson Scheme

The explicit schemes of the Runge–Kutta type (Jameson *et al.*, 1981) for the numerical integration of the Euler equations have a larger stability interval than many previous explicit schemes. In addition, the stability robustness of the Runge–Kutta-type schemes (we will also call them the Jameson's schemes) increases with the number of stages (Ganzha and Vorozhtsov, 1996b). Along with this property the Jameson's schemes possess a number of other positive properties (Ganzha and Vorozhtsov, 1993). This has caused

a very wide acceptance of the Runge–Kutta-type schemes for the numerical solution of the aerodynamics problems.

From the viewpoint of complexity, the Runge–Kutta-type schemes represent a very interesting object on which one can validate the efficiency of the new approaches to the symbolic–numeric stability investigation of difference schemes. The complexity of the step operator S in (1) increases in these schemes non-linearly with the number of stages. This leads to a corresponding increase of the demands on the computer storage and speed of the processing of symbolic expressions.

The three-stage Runge–Kutta-type schemes may be considered as the schemes possessing already a larger complexity than other well-known previous schemes (for example, the MacCormack scheme, which was very popular in the past, contains only two stages; the "predictor" stage and the "corrector" stage (see Ganzha and Vorozhtsov, 1996a). The number of stages, three altogether, ensures a good stability robustness. In this connection, the three-stage schemes are widely used in aerodynamic computations (see a review of the relevant works in Ganzha and Vorozhtsov, 1993).

In the original work (Jameson *et al.*, 1981) the Runge–Kutta-type schemes were implemented within the framework of the finite-volume method whose application enables one to avoid the use of the Euler equations in the curvilinear coordinates. It is sufficient to take a small control volume at the center of which the current node (x_{jk}, y_{jk}) of the curvilinear computing mesh is located, and to transform the spatial differentiation operators in the Euler equations with the aid of the Green's theorem.

We demonstrate this procedure in example of the Euler equations governing the two-dimensional non-stationary flow of an inviscid compressible non-heat-conducting gas:

$$\frac{\partial \vec{u}}{\partial t} + \frac{\partial \vec{F}}{\partial x} + \frac{\partial \vec{G}}{\partial y} = 0, \tag{1}$$

where t is the time, x, y are the spatial Cartesian coordinates, and the flux vectors $\vec{F}$ and $\vec{G}$ are as follows:

$$\vec{F}(\vec{u}) = (\rho u, p + \rho u^2, \rho uv, pu + \rho uE)^T, \qquad \vec{G}(\vec{u}) = (\rho v, \rho uv, p + \rho v^2, pv + \rho vE)^T,$$
$$\vec{u} = (\rho, \rho u, \rho v, \rho E)^T. \tag{2}$$

The system (1)–(2) is completed in the following by the ideal gas equation of state

$$p = (\gamma - 1)\rho\epsilon, \tag{3}$$

where γ is the ratio of the gas specific heats. In equations (1)–(3), ρ is the fluid density, u, v are the fluid velocity components along the x- and y-axis, $E = \varepsilon + (u^2 + v^2)/2$, ε is the specific internal energy, and p is the pressure.

Let us choose an arbitrary cell and denote its four vertices (see Figure 2)

$$P_1(x_{j,k}, y_{j,k}), \quad P_2(x_{j+1,k}, y_{j+1,k}), \quad P_3(x_{j+1,k+1}, y_{j+1,k+1}), \quad P_4(x_{j,k+1}, y_{j,k+1}).$$

Let us take the region bounded by the contour $\Gamma_{j,k} = P_1P_2P_3P_4P_1$ as the control volume $V_{j,k}$. Let $S_{j,k}$ be the area of the region bounded by the contour $\Gamma_{j,k}$. To derive the approximate formulas of the finite-volume method we use Green's theorem:

$$\int\int_{V_{j,k}} \left(\frac{\partial \vec{u}}{\partial t} + \frac{\partial \vec{F}}{\partial x} + \frac{\partial \vec{G}}{\partial y} \right) dxdy = \int\int_{V_{j,k}} \frac{\partial \vec{u}}{\partial t} dxdy + \oint_{\Gamma_{j,k}} (\vec{F}dy - \vec{G}dx) = 0. \tag{4}$$

One can evaluate the first integral in the right-hand side of (4) with the aid of the

mean-value theorem:

$$\int\int_{V_{j,k}} \frac{\partial \vec{u}}{\partial t} dxdy \approx S_{j,k} \left(\frac{\partial \vec{u}}{\partial t} \right)_{j,k},$$

where the subscripts (j,k) refer to the control volume center $V_{j,k}$; that is the components of the vector $\vec{u}$ are computed at the center of the volume $V_{j,k}$. Then we obtain instead of (4) the following approximate equality:

$$\left(\frac{d\vec{u}}{dt} \right)_{j,k} + Pu(t) = 0, \tag{5}$$

where

$$Pu(t) = \frac{1}{S_{j,k}} \oint_{\Gamma_{j,k}} (\vec{F}dy - \vec{G}dx). \tag{6}$$

There are two schemes for the approximation of the integrals in the finite-volume method: the centered scheme and the nodal scheme (Fletcher, 1996). We use the centered scheme in the following (see Figure 2), that is we will assume that the components of the vector $\vec{u}$ are computed at the centers of the curvilinear grid cells. The area $S_{j,k}$ can be computed within the framework of this scheme with the use of the notations of Figure 2 as

$$\begin{aligned} S_{j,k} &= \int\int_{V_{j,k}} dxdy = \oint_{\Gamma_{j,k}} xdy \\ &= x_1(y_{P_2} - y_{P_1} + x_2(y_{P_3} - y_{P_2}) + x_3(y_{P_4} - y_{P_3}) + x_4(y_{P_1} - y_{P_4}) \\ &= \frac{1}{2}[(x_{j+1,k+1} - x_{j,k})(y_{j,k+1} - y_{j+1,k}) + (x_{j+1,k} - x_{j,k+1})(y_{j+1,k+1} - y_{j,k})]. \end{aligned} \tag{7}$$

The fluxes $\vec{F}(\vec{u})$ and $\vec{G}(\vec{u})$ across the boundaries of the control volume $V_{j,k}$ are computed as the mean values. For example,

$$(\rho u)_2 = \frac{1}{2}[(\rho u)_{j,k} + (\rho u)_{j+1,k}]. \tag{8}$$

As was noted in Fletcher (1996) the use of the approximation of form (8) for fluxes leads in the particular case of a rectangular uniform grid to a difference scheme with central differences. Therefore, the above scheme of the finite-volume method has the second order of approximation in spatial variables under the condition that the curvilinear grid is sufficiently smooth (see also Ganzha and Vorozhtsov, 1996b).

Denote by $(P_h\vec{u})_{j,k}$ the approximation of integral (6) using the formulas of type (8). In the case where no artificial dissipators are used the operator $(P_h\vec{u})_{j,k}$ has the form

$$\begin{aligned} (P_h\vec{u})_{j,k} = \frac{1}{2S_{j,k}}[&(\vec{F}_{j,k} + \vec{F}_{j,k-1})(y_{j+1,k} - y_{j,k}) + (\vec{F}_{j,k} + \vec{F}_{j+1,k})(y_{j+1,k+1} - y_{j+1,k}) \\ &+(\vec{F}_{j,k} + \vec{F}_{j,k+1})(y_{j,k+1} - y_{j+1,k+1}) + (\vec{F}_{j,k} + \vec{F}_{j-1,k})(y_{j,k} - y_{j,k+1}) \\ &-(\vec{G}_{j,k} + \vec{G}_{j,k-1})(x_{j+1,k} - x_{j,k}) - (\vec{G}_{j,k} + \vec{G}_{j+1,k})(x_{j+1,k+1} - x_{j+1,k}) \\ &-(\vec{G}_{j,k} + \vec{G}_{j,k+1})(x_{j,k+1} - x_{j+1,k+1}) - (\vec{G}_{j,k} + \vec{G}_{j-1,k})(x_{j,k} - x_{j,k+1})]. \end{aligned} \tag{9}$$

Let us approximate the system of ordinary differential equations (5) by the three-stage

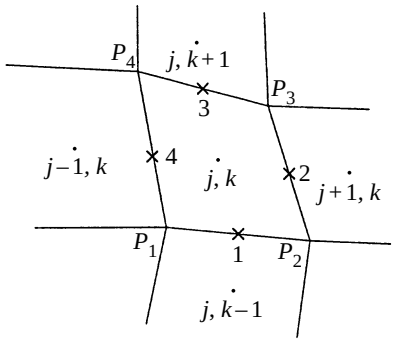

Figure 2. The centered scheme.

Runge–Kutta scheme of the form (Jameson and Schmidt, 1985)

$$\vec{u}^{(0)} = \vec{u}^n; \quad \vec{u}^{(1)} = \vec{u}^0 - \alpha_1 \tau P_h \vec{u}^{(0)}; \quad \vec{u}^{(2)} = \vec{u}^0 - \alpha_2 \tau P_h \vec{u}^{(1)};$$
$$\vec{u}^{(3)} = \vec{u}^0 - \tau P_h \vec{u}^{(2)}; \quad \vec{u}^{n+1} = \vec{u}^{(3)}, \tag{10}$$

where τ is the time step, α_1 and α_2 are the constant weight parameters. The parameters α_1 and α_2 can be chosen from the requirement of stability robustness (Jameson and Schmidt, 1985). For example, the choice $\alpha_1 = \alpha_2 = \frac{1}{2}$ ensures the stability condition $0 \le |a\tau/h| \le 2$ in the case where scheme (10) is applied for the difference approximation of the scalar advection equation $u_t + au_x = 0$ (Ganzha and Vorozhtsov, 1993). We will use the values $\alpha_1 = \alpha_2 = \frac{1}{2}$ in what follows.

The Fourier method is applicable only to the linear difference schemes with constant coefficients. The difference equations (10) are non-linear, therefore, we have to linearize them before the application of the Fourier method. For this purpose we use the formulas

$$\vec{F}_{m,q} \approx \vec{F}(\vec{u}_{j,k}) + A(\vec{a}_{j,k})\delta\vec{u}_{m,q}; \qquad \vec{G}_{m,q} \approx \vec{G}(\vec{u}_{j,k}) + B(\vec{a}_{j,k})\delta\vec{u}_{m,q}, \tag{11}$$

where $\delta\vec{u}_{m,q} = \vec{u}_{m,q} - \vec{u}_{j,k}, (m,q)$ are any values of the subscripts entering (10); the Jacobi matrices A and B are defined by formulas

$$A = \partial\vec{F}(\vec{u})/\partial\vec{u}, \qquad B = \partial\vec{G}(\vec{u})/\partial\vec{u}. \tag{12}$$

Denote the linearization of the operator $P_h\vec{u}$ by $\bar{P}_h\vec{u}$. Substituting formulas (11) in (10) we obtain the following expression for the operator $(\bar{P}_h\vec{u})_{j,k}$:

$$\begin{aligned}(\bar{P}_h\vec{u})_{j,k} = \frac{1}{2S_{j,k}}[&A\vec{u}_{j,k-1}(y_{j+1,k} - y_{j,k}) + A\vec{u}_{j+1,k}(y_{j+1,k+1} - y_{j+1,k})\\ &+A\vec{u}_{j,k+1}(y_{j,k+1} - y_{j+1,k+1}) + A\vec{u}_{j-1,k}(y_{j,k} - y_{j,k+1})\\ &-B\vec{u}_{j,k-1}(x_{j+1,k} - x_{j,k}) - B\vec{u}_{j+1,k}(x_{j+1,k+1} - x_{j+1,k})\\ &-B\vec{u}_{j,k+1}(x_{j,k+1} - x_{j+1,k+1}) - B\vec{u}_{j-1,k}(x_{j,k} - x_{j,k+1})].\end{aligned} \tag{13}$$

For the sake of brevity we have used here the notations $A = A(\vec{u}_{j,k})$, $B = B(\vec{u}_{j,k})$. The linearization assumes that the coefficients A and B are constant. Then we can see from (13) that the operator $\bar{P}_h$ is linear. A direct substitution of expression (13) in the three-stage scheme (10) leads to huge algebraic expressions at the stage of a symbolic computation of the Fourier symbol G. However, it is possible to simplify significantly the expression for the operator $\bar{P}_h$ under certain reasonable assumption. Rewrite expres-

sion (13) in the form

$$(\bar{P}_h\vec{u})_{j,k} = A\Lambda_1\vec{u}_{j,k} + B\Lambda_2\vec{u}_{j,k}, \tag{14}$$

where

$$\begin{aligned}\Lambda_1\vec{u}_{j,k} = \frac{1}{2S_{j,k}}[&(y_{j+1,k} - y_{j,k})\vec{u}_{j,k-1} + (y_{j+1,k+1} - y_{j+1,k})\vec{u}_{j+1,k}\\ &+(y_{j,k+1} - y_{j+1,k+1})\vec{u}_{j,k+1} + (y_{j,k+1})\vec{u}_{j-1,k}],\end{aligned} \tag{15}$$

$$\begin{aligned}\Lambda_2\vec{u}_{j,k} = \frac{1}{2S_{j,k}}[&(x_{j,k} - x_{j+1,k})\vec{u}_{j,k-1} + (x_{j+1,k} - x_{j+1,k+1})\vec{u}_{j+1,k}\\ &+(x_{j+1,k+1} - x_{j,k+1})\vec{u}_{j,k+1} + (x_{j,k+1} - x_{j,k})\vec{u}_{j-1,k}].\end{aligned} \tag{16}$$

Assume that a specific control volume $V_{j,k}$ is such that the following relations are satisfied:

$$\begin{aligned}&y_{j+1,k+1} - y_{j,k+1} = y_{j+1,k} - y_{j,k}; \qquad y_{j+1,k+1} - y_{j+1,k} = y_{j,k+1} - y_{j,k};\\ &x_{j+1,k+1} - x_{j+1,k} = x_{j,k+1} - x_{j,k}; \qquad x_{j+1,k+1} - x_{j,k+1} = x_{j+1,k} - x_{j,k}.\end{aligned} \tag{17}$$

It follows from Figure 2 that conditions (17) are the conditions that the pairs of the sides P_3P_4 and P_1P_2, P_2P_3 and P_4P_1 are parallel. Thus, the volume $V_{j,k}$ proves to be a parallelogram under conditions (17). Note that these conditions are satisfied for all cells of a parallelogram grid, which is a particular case of the curvilinear grid (Thompson *et al.*, 1985; Ganzha and Vorozhtsov, 1996a). In the case of a general curvilinear grid only some cells will have the parallelogram form or will differ little from the parallelograms.

Expression (13) simplifies greatly under satisfaction of conditions (17):

$$(\bar{P}_h\vec{u})_{j,k} = A_1(\vec{u}_{j,k}, x, y)\frac{\vec{u}_{j+1,k} - \vec{u}_{j-1,k}}{2} + A_2(\vec{u}_{j,k}, x, y)\frac{\vec{u}_{j,k+1} - \vec{u}_{j,k-1}}{2}, \tag{18}$$

where

$$\begin{aligned}A_1 &= \frac{1}{S_{j,k}}[(y_{j,k+1} - y_{j,k})A - (x_{j,k+1} - x_{j,k})B],\\ A_2 &= \frac{1}{S_{j,k}}[(x_{j+1,k} - x_{j,k})B - (y_{j+1,k} - y_{j,k})A].\end{aligned} \tag{19}$$

The linearized scheme corresponding to scheme (10) has the form

$$\begin{aligned}&\vec{u}^{(0)} = \vec{u}^n; \quad \vec{u}^{(1)} = \vec{u}^0 - \alpha_1\tau\bar{P}_h\vec{u}^{(0)}; \quad \vec{u}^{(2)} = \vec{u}^0 - \alpha_2\tau\bar{P}_h\vec{u}^{(1)};\\ &\vec{u}^{(3)} = \vec{u}^0 - \tau\bar{P}_h\vec{u}^{(2)}; \quad \vec{u}^{n+1} = \vec{u}^{(3)}.\end{aligned} \tag{20}$$

Eliminating the intermediate quantities $\vec{u}^{(1)}$ and $\vec{u}^{(2)}$ from (20) we obtain the two-level difference scheme (1), where the step operator S has the form

$$S(x, y, \vec{u}_0) = I - \tau\bar{P}_h + \alpha_2(\tau\bar{P}_h)^2 - \alpha_1\alpha_2(\tau\bar{P}_h)^3, \tag{21}$$

and $\vec{u}_0$ is the "frozen" solution vector whose components enter the Jacobi matrices (12). Let us substitute into (1) the solution of the form

$$\vec{u}^n_{j,k} = \vec{u}_0 e^{i(k_1 j + k_2 k - \omega n\tau)},$$

where k_1 and k_2 are the real wavenumbers, and ω is the wave frequency. Then we obtain from (21) the Fourier symbol for any curvilinear grid:

$$G = I + Z + \alpha_2 Z^2 + \alpha_1\alpha_2 Z^3, \tag{22}$$

where Z is the Fourier symbol of operator $\bar{P}_h$,

$$Z = -i\tau(\sin k_1 A_1 + \sin k_2 A_2). \tag{23}$$

Let $\mu_m, m = 1, 2, 3, 4$ be the eigenvalues of the matrix Z. Then the eigenvalues $\lambda_m, m = 1, 2, 3, 4$, of the matrix G (22) have the form

$$\lambda_j = 1 + \mu_j + \alpha_2\mu_j^2 + \alpha_1\alpha_2\mu_j^3, \qquad j = 1, 2, 3, 4. \tag{24}$$

The matrix Z represents a linear combination of the Jacobi matrices A and B:

$$Z = d_1 A + d_2 B, \tag{25}$$

where with regard for (19)

$$\begin{aligned} d_1 &= -\frac{i\tau}{S_{j,k}}[(\sin k_1)(y_{j,k+1} - y_{j,k}) - (\sin k_2)(y_{j+1,k} - y_{j,k})], \\ d_2 &= -\frac{i\tau}{S_{j,k}}[-(\sin k_1)(x_{j,k+1} - x_{j,k}) + (\sin k_2)(x_{j+1,k} - x_{j,k})]. \end{aligned} \tag{26}$$

The eigenvalues μ_j of the matrix (25) are known to have the form (Warming *et al.*, 1975)

$$\mu_1 = \mu_2 = d_1 u + d_2 v, \qquad \mu_3 = \mu_1 + c(d_1^2 + d_2^2)^{0.5}, \qquad \mu_4 = \mu_1 - c(d_1^2 + d_2^2)^{0.5}, \tag{27}$$

where c is the velocity of sound in gas.

Thus, we have found the analytical expressions for the eigenvalues $\lambda_1, \ldots, \lambda_4$ of the Fourier symbol G (22) of the Runge–Kutta scheme (20) on a curvilinear grid. This has become possible owing to the fact that the matrix G is a polynomial in some other matrix Z (22). As we have shown in (Ganzha and Vorozhtsov, 1996a), many difference schemes for the Euler equations do not possess this property, and the expressions for λ_j cannot then be found in analytic form. In addition, in the case of difference schemes approximating the Navier–Stokes equations for a compressible fluid, there is no similarity matrix T diagonalizing both the Jacobi matrices A and B and the matrices arising from the approximation of viscous terms (Warming *et al.*, 1975). For such cases we have proposed in Ganzha and Vorozhtsov (1996a) several strategies of symbolic–numeric stability analysis.

In the next Section we propose a new symbolic–numeric algorithm for the stability analysis of difference schemes on curvilinear grids. This algorithm is far superior in its speed and accuracy to any of the symbolic–numeric methods based on the Fourier method that was presented in Ganzha and Vorozhtsov (1996a).

For the validation of our symbolic–numeric method we will use the analytical von Neumann stability condition, which can be obtained by using the analytic expressions (24), (26) and (27). In order to obtain this condition let us at first introduce the square region

$$\Pi = \{(k_1, k_2) | 0 \le k_l \le 2\pi, l = 1, 2\}. \tag{28}$$

The value 2π represents here the period of the entries of the amplification matrix G (22) in k_1 and k_2. The coefficients of the characteristic equation $\det(\lambda I - G) = 0$ may have another period, which however does not exceed the period of the entries of G, because these coefficients are polynomials in the entries of G.

It follows from (26) that all the eigenvalues $\mu_j, j = 1, \ldots, 4$, are purely imaginary. For this case, the following optimal values of the weight parameters α_1 and α_2 were indicated in Pike and Roe (1985) for the three-stage Jameson's scheme (20):

$$\alpha_1 = 0.5, \qquad \alpha_2 = 0.5. \tag{29}$$

We will use these values in the following. Other choices for α_1 and α_2 from the requirement of increased stability robustness were considered in Jameson and Schmidt (1985) and Ganzha and Vorozhtsov (1996a). Then the von Neumann stability condition $|\lambda_j| \leq 1, j = 1, \ldots, 4$ implies, referring to (3),

$$\tau \cdot \max_{\vec{k} \in \Pi} [\varphi^+(\xi_x, \xi_y, \eta_x, \eta_y, \vec{u}_0, \vec{k}), \varphi^-(\xi_x, \xi_y, \eta_x, \eta_y, \vec{u}_0, \vec{k})] \leq C, \tag{30}$$

where $\vec{k} = (k_1, k_2)$,

$$\varphi^{\pm}(\xi_x, \xi_y, \eta_x, \eta_y, \vec{u}_0, \vec{k}) = |\xi_x u \sin k_1 + \eta_x u \sin k_2 + \xi_y v \sin k_1 + \eta_y v \sin k_2 \pm [(c\xi_x \sin k_1 + c\eta_x \sin k_2)^2 + (c\xi_y \sin k_1 + c\eta_y \sin k_2)^2]^{0.5}|, \tag{31}$$

C is the Courant number; $C = 2$ in the case of the three-stage scheme (20) and $\alpha_1 = \alpha_2 = 0.5$. We have introduced in (30) the notation

$$\begin{aligned} \xi_x &= \frac{y_{j,k+1} - y_{j,k}}{S_{j,k}}, \qquad \eta_x = -\frac{y_{j+1,k} - y_{j,k}}{S_{j,k}}, \\ \xi_y &= -\frac{x_{j,k+1} - x_{j,k}}{S_{j,k}}, \qquad \eta_y = \frac{x_{j+1,k} - x_{j,k}}{S_{j,k}}. \end{aligned} \tag{32}$$

One can show that formulas (32) represent the difference approximations of the derivatives $\partial\xi/\partial x$, $\partial\eta/\partial x$, $\partial\xi/\partial y$, $\partial\eta/\partial y$ (Thompson *et al.*, 1985).

It can be shown (see also Pike and Roe, 1985; Ganzha and Vorozhtsov, 1993) that the left-hand side of the von Neumann stability condition (30) remains the same for Runge–Kutta-type schemes with a different number m of intermediate stages ($m \geq 1$). Only the value of the Courant number C on the right-hand side of (30) is different for different Runge–Kutta schemes in the absence of added artificial dissipators; for example, $C = 4$ for the optimal five-stage Runge–Kutta scheme. Therefore, the subsequent conclusions on the effect of curvilinear grid topology on the stability of scheme (20) remain valid also for other explicit Runge–Kutta schemes (i.e. for other numbers of stages m).

It is easy to show (cf. Vorozhtsov, 1995; Ganzha and Vorozhtsov, 1996a) that the functions $\tau\varphi^{\pm}(\cdot)$ defined by (31) involve the following non-dimensional variables:

$$\begin{aligned} \kappa_1 &= c\tau\xi_x, \qquad \kappa_2 = u\tau\xi_x, \qquad \kappa_3 = v\tau\xi_x, \\ \kappa_4 &= \eta_x/\xi_x, \qquad \kappa_5 = \eta_y/\xi_x, \qquad \kappa_6 = \xi_y/\xi_x. \end{aligned} \tag{33}$$

We can rewrite equation (31) in terms of the introduced variables (33) as follows:

$$\tau\varphi^{\pm}(\vec{\kappa}, \vec{k}) = |\kappa_2 \sin k_1 + \kappa_2 \sin k_2 + \kappa_3\kappa_6 \sin k_1 + \kappa_3\kappa_5 \sin k_2 \pm [(\kappa_1 \sin k_1 + \kappa_1\kappa_4 \sin k_2)^2 + (\kappa_1\kappa_6 \sin k_1 + \kappa_1\kappa_5 \sin k_2)^2]^{0.5}|, \tag{34}$$

where $\vec{\kappa} = (\kappa_1, \kappa_2, \kappa_3, \kappa_4, \kappa_5, \kappa_6)$. Since $\max_{k_m} |\kappa_j \sin k_m| = |\kappa_j|, j = 1, \ldots, 6, k_m \in \Pi, m = 1, 2$, it follows from (34) that the local von Neumann necessary stability condition of the linearized Jameson's scheme (20) on a parallelogram grid has the form

$$|\kappa_2| + |\kappa_2\kappa_4| + |\kappa_3\kappa_6| + |\kappa_3\kappa_5| + [(|\kappa_1| + |\kappa_1\kappa_4|)^2 + (|\kappa_1\kappa_6| + |\kappa_1\kappa_5|)^2]^{0.5} \leq C. \tag{35}$$

It follows from (35) that the functional form of the necessary stability condition of Jameson's schemes does not depend on the specific grid density, and in this sense condition (35) is universal. However, the local time steps $\tau_{j,k} = \tau(x_{jk}, y_{jk})$ in a specific grid node (x_{jk}, y_{jk}) will of course depend on grid density in the neighborhood of a given grid

node, because in accordance with (35) and (33)

$$\tau_{jk}^{n+1} \le \theta C/\{|u_{jk}^n(\xi_x)_{jk}| + |u_{jk}^n(\eta_x)_{jk}| + |v_{jk}^n(\xi_y)_{jk}| + |v_{jk}^n(\eta_y)_{jk}| + c_{jk}^n[(|(\xi_x)_{jk}| + |(\eta_x)_{jk}|)^2 + (|(\xi_y)_{jk}| + |(\eta_y)_{jk}|)^2]^{0.5}\}. \tag{36}$$

In accordance with the foregoing (see the discussion of relations (17)) the stability condition (35) can be considered as an approximate stability condition on a general curvilinear grid. For the general grid it is reasonable to replace the constant C in (35) with θC, where θ is a safety factor, $0 < \theta \le 1$.

Let us now consider also the particular case of a uniform rectangular spatial grid:

$$x_{j+1,k} = x_{j,k} + h_1, \qquad x_{j,k+1} = x_{j,k}, \qquad y_{j+1,k} = y_{j,k}, \qquad y_{j,k+1} = y_{j,k} + h_2 \qquad \forall j,k, \tag{37}$$

where h_1 and h_2 are the grid steps along the x- and y-axes, respectively. We now look at the form of the variables (33) in this particular case. For this purpose we substitute (37) in (32). As a result we obtain:

$$\kappa_1 = \frac{c\tau}{h_1}, \qquad \kappa_2 = \frac{u\tau}{h_1}, \qquad \kappa_3 = \frac{v\tau}{h_1}, \qquad \kappa_4 = 0, \qquad \kappa_5 = \frac{h_1}{h_2}, \qquad \kappa_6 = 0.$$

Therefore, the general necessary stability condition (35) simplifies for the three-stage scheme (20) to the condition

$$|\kappa_2| + |\kappa_3|\kappa_5 + \kappa_1\sqrt{1+\kappa_5^2} \le 2. \tag{38}$$

At the specified cell aspect ratio κ_5 the equality

$$\kappa_1 = \frac{2 - |\kappa_2| - |\kappa_3|\kappa_5}{\sqrt{1+\kappa_5^2}} \tag{39}$$

thus determines a surface $\kappa_1 = f(\kappa_2, \kappa_3)$ of the necessary stability region of Jameson's scheme (20) on a uniform rectangular grid. In particular, at the apex of the pyramid (39), where $\kappa_2 = \kappa_3 = 0$, we have

$$\kappa_1 = \frac{2}{\sqrt{1+\kappa_5^2}} \tag{40}$$

(see also Figure 3).

6. Implementation in *Mathematica*

6.1. SYMBOLIC STAGES

The feature of the symbolic–numeric algorithm proposed by Ganzha *et al.* (1992) is that it makes use of the Newton identities to compute the numerical values of the coefficients $c_j(\vec{\kappa}, \vec{k})$ of the characteristic equation

$$\det(\lambda I - G) = \sum_{j=0}^{4} c_j(\vec{\kappa}, \vec{k})\lambda^{4-j} = 0, \tag{1}$$

where $\vec{\kappa} = (\kappa_1, \ldots, \kappa_M)$, $M \ge 1$ is the number of non-dimensional similarity parameters $\kappa_1, \ldots, \kappa_M$ like the parameters (33). Thus, the symbolic computations were used in Ganzha *et al.* (1992) beginning from the initial stage of deriving the difference scheme in integral steps "n" and "$n+1$" and ending with the symbolic computation of the entries

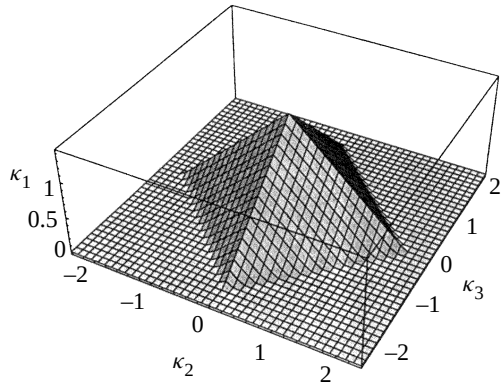

Figure 3. The necessary stability region of Jameson's scheme (20) on an uniform rectangular grid.

g_{jk}, $j,k = 1,\ldots,4$, of the amplification matrix G. A further computation proceeded as a numerical computation in the arithmetic of the machine floating-point numbers using a FORTRAN code, so that the coefficients c_j in (1) were computed numerically. In this way we avoided the tremendous expression swell, which would take place in the case of symbolic computation of the coefficients c_j in (1). This strategy of terminating the symbolic mode of computation at the stage of symbolic computation of the entries of G appeared attractive for us also in the case of difference schemes on curvilinear grids. However, an attempt to use the symbolic stages from Ganzha *et al.* (1992) resulted in the memory exhaustion already at the initial stage of deriving the difference equations in integral steps from the three-stage difference scheme (20) by eliminating the intermediate values $\vec{u}^{(1)}$ and $\vec{u}^{(2)}$. The main reason for this was the fact that the symbolic method of Ganzha *et al.* (1992) requires the explicit specification of the entries of the Jacobi matrices A and B (12) already at the early stages of symbolic computations. In the case of curvilinear grid one has to use much more complex matrices A_1 and A_2 (19) in accordance with (18), which immediately leads to memory exhaustion.

In this connection we had to search for other way to circumvent this problem. Since the amplification matrix G is a function of the Jacobi matrices A and B for many difference schemes for Euler equations, there emerges the idea of working with A and B as matrix objects, which are however not specified explicitly as 4×4 matrices at the early stages of symbolic computations. In the process of such computations the expressions of the form

$$c_1AB + c_2BA, \quad c_1ABB + c_2BAB, \tag{2}$$

etc. arise (see equations (18), (19) and (21)). It is well known (Ganzha and Vorozhtsov, 1996a) that the gas dynamical matrices A and B do not commute, therefore, the conventional commutative multiplication is inapplicable. *Mathematica*, however, treats A and B in (2) as commutative quantities and transforms them into $(c_1+c_2)AB$ and $(c_1+c_2)AB^2$, which leads to incorrect results in our case.

There is a non-commutative algebra in the CAS REDUCE 3.6 (Hearn, 1995): if we write

```
noncom A,B
```

then the expressions `A(x)*A(y)-A(y)*A(x)` or `A(x)*B(y)-B(y)*A(x)` are not equal to

zero. The only limitation for A and B consists here of the fact that the A and B should be the operators rather than variables. It is easy to take this limitation into account in our case, since the elements of the Jacobi matrices `A` and `B` depend on several variables.

There is no possibility in CAS *Mathematica* for introducing the declaration of the type `non-commutative A,B`, where the `A`, `B` are variables. In order to solve this problem we have proposed the idea of replacing the commutative multiplication operation `Times` in *Mathematica* with the non-commutative operation `Dot`. To illustrate the implimentation of this idea in a *Mathematica* program let us assume that certain quantities `A1` and `A2` do not commute and consider the following program:

```
c1 = HoldForm[(A1+A2) (A1+A2) (A1+A2)]; d1 = c1/.{Plus->List, Times->Tim};
d2 = d1/.HoldForm[x_[y___]]->HoldForm[x,y]; e = d2/.HoldForm->Outer;
f  = e/.List->Plus; pu3=f/. Tim[x_,y__,z_]->Dot[x,y,z]
```

This program computes the product $(A1 + A2)(A1 + A2)(A1 + A2)$ in the correct form, which takes into account the non-commutativity of the quantities $A1$ and $A2$:

```
A1 . A1 . A1 + A1 . A1 . A2 + A1 . A2 . A1 +A1 . A2 . A2 + A2 .
A1 . A1 +
A2 . A1 . A2 + A2 . A2 . A1 + A2 . A2 . A2
```

The above program fragment is a key element of our *Mathematica* program, which has enabled us to obtain the amplification matrix B symbolically as a function of the matrices A and B, although these matrices themselves were not specified explicitly as matrices. Let us now describe the sequential stages of our symbolic algorithm whose output is the amplification matrix G of a given difference scheme.

Step 6.1. Elimination of the intermediate values $\vec{u}^{(1)}$ and $\vec{u}^{(2)}$ from the difference equation $\vec{u}^{n+1} - \vec{u}^{(3)} = 0$ of scheme (20). Let us introduce in the *Mathematica* program the notations `uv[1,n_,i_,j]` $= \vec{u}^{(1)}_{ij}$, `uv[2,n_,i_,j_]` $= \vec{u}^{(2)}_{ij}$ and the functions

```
uv[1,n_,i_,j_]:=uv0[n,i,j]-alf1 dt pu*uv[0,n,i,j]
uv[2,n_,i_,j_]:=uv0[n,i,j]-alf2 dt pu*uv[1,n,i,j]
uv[3,n_,i_,j_]:=uv0[n,i,j]-dt pu*uv[2,n,i,j]
```

where `pu` stands for the matrix difference operator $\bar{P}_h$; `dt` $= \tau$, `alf1` $= \alpha_1$, `alf2` $= \alpha_2$. We note that the quantity `uv0[n,i,j]` $= \vec{u}^n_{ij}$ is not declared here as a vector of four components. We have denoted the result of the action of the difference operator $\bar{P}_h$ on the grid function $\vec{u}^n_{ij}$ simply as a usual product: $\bar{P}_h\vec{u}^n_{ij}$ = `pu*uv0[n,i,j]`. As we have shown in Ganzha and Vorozhtsov (1996b), such difference operators can be efficiently implemented in *Mathematica* with the aid of transformation rules. The elimination of the values $\vec{u}^{(1)}$ and $\vec{u}^{(2)}$ is performed in our program with the aid of the command

```
sch=ID uv[3,n,i,j]/. {uv[0,n,i_,j_]->uv0[n,i,j]}//Expand
```

where `ID` is assumed to be the 4×4 identity matrix; it will be specified explicitly as a

matrix at a much later stage. The execution of the above command enables us to obtain the left-hand side of the two-level difference equation $\vec{u}^{n+1} - \vec{u}^{(3)} = 0$:

```
                                                        2     2
ID uv0[n, i, j] - dt ID pu uv0[n, i, j] + alf2 dt ID pu uv0[n, i, j] -
             3      3
alf1 alf2 dt ID pu  uv0[n, i, j]
```

Comparing this expression with equations (1) and (21) we can see that our *Mathematica* program produces a correct result.

Step 6.2. Substitution of expression (18) for the difference operator $\bar{P}_h$ into the difference scheme in integral steps. Let us introduce the shift operators

$$T_1^m u(x_{j,k}, y_{j,k}, t^n) = u(x_{j+m,k}, y_{j+m,k}, t^n);$$
$$T_2^m u(x_{j,k}, y_{j,k}, t^n) = u(x_{j,k+m}, y_{j,k+m}, t^n).$$

Since the operator (18) involves central differences we can rewrite it in the form $\bar{P}_h = A_1\Delta_1 + A_2\Delta_2$, where

$$\Delta_1 = (T_1^1 - T_1^{-1})/2, \qquad \Delta_2 = (T_2^1 - T_2^{-1})/2. \tag{3}$$

Substituting the expressions for A_1 and A_2 (19) we can write the operator $\bar{P}_h$ as

$$\bar{P}_h = A\Lambda_1 + B\Lambda_2,$$

where the difference operators Λ_1 and Λ_2 have with regard for (32) the form

$$\Lambda_1 = \xi_x\Delta_1 + \eta_x\Delta_2, \qquad \Lambda_2 = \xi_y\Delta_1 + \eta_y\Delta_2. \tag{4}$$

It is easy to show that the operators Λ_1 and Λ_2 commute at the frozen coefficients ξ_x, η_x, ξ_y, η_y. The repeated use of the above-discussed implementation of non-commutative multiplication with *Mathematica* for the computation of the powers $\bar{P}_h^2$, $\bar{P}_h^3$ enables us to compute symbolically the matrix expression for the step operator $S(x, y, \vec{u}_0)$ as the right-hand side of (1):

```
ID uv0[n, i, j] - dt (A t1 + B t2) uv0[n, i, j] +
   2     2                                      2
(dt    (t1 A . A + t1 t2 A . B + t1 t2 B . A + t2 B . B) uv0[n, i, j]) /
2 -    3     3              2                 2
(dt   (t1 A . A . A + t1 t2 A . A . B + t1 t2 A . B . A +  t1 t2 A . B .
 B +   2                   2                 2              3
t1 t2 B . A . A + t1 t2 B . A . B + t1 t2 B . B . A + t2 B . B . B)
uv0[n, i, j]) / 4
```

where `t1` $= \Lambda_1$, `t2` $= \Lambda_2$.

Step 6.3. Fourier transform of the step operator $S(x, y, \vec{u}_0)$. The next step would be the substitution of the expressions (4) for Λ_1 and Λ_2 into the obtained expression for $S(\cdot)$.

However, this leads to a very big algebraic expression. For example, the product $\Lambda_1\Lambda_2$ has the form

$$\Lambda_1\Lambda_2 = \xi_x\xi_y\Delta_1^2 + \xi_x\eta_y\Delta_1\Delta_2 + \eta_x\xi_y\Delta_2\Delta_1 + \eta_x\eta_y\Delta_2^2.$$

We should further substitute the definitions for Δ_1 and Δ_2 into this expression. An attempt at implementing this chain of substitutions resulted in the memory expenses of about 23 MB and in several hours of CPU time on a Silicon Graphics, Inc. (SGI) 200 MHz with 264 MB.

In this connection we have used another possibility for the symbolic computation of the Fourier transform of the step operator $S(x, y, \vec{u}_0)$. Since S is a linear difference operator, we can at first compute the Fourier transform of each item entering S. Denote by $\mathcal{F}(\Lambda_m)$ the Fourier symbol of the difference operator Λ_m, $m = 1, 2$. It is easy to calculate by hand that

$$\mathcal{F}(\Lambda_1) = i\xi_x \sin k_1 + i\eta_x \sin k_2, \qquad \mathcal{F}(\Lambda_2) = i\xi_y \sin k_1 + i\eta_y \sin k_2. \tag{5}$$

Therefore, we can substitute these expressions into the above expression for $S(\cdot)$:

```
t1=I sa ksix + I sb etax; t2=I sa ksiy + I sb etay; sch4=sch4
```

As a result we obtain the amplification matrix G as a function of A, B, k_1, k_2; in the above program fragment, `sa` $= \sin k_1$, `sb` $= \sin k_2$, `ksix` $= \xi_x$, `etax` $= \eta_x$, etc.

This symbolic computation has required only a few minutes of CPU time on the same machine. We do not present here the expression for G because of its relatively big size of about 40 lines of text.

Step 6.4. Symbolic computation of the entries of G. To save the needed memory at this stage, we at first substitute the entries of the matrices A and B as `a21` $= a_{2,1}$, `b43` $= b_{4,3}$, etc., where we have denoted by $a_{j,k}$ and $b_{j,k}$; $j, k = 1, \ldots, 4$ the entries of the matrices A and B, respectively. After that we find the expressions for the first row of G by replacing `ajk` with their specific entries corresponding to the gas dynamic Jacobi matrices A and B (the expressions for these entries may be found in Ganzha and Vorozhtsov, 1996a).

The first row elements are thus the functions of the components of the solution vector $\vec{u}$ in (2) and of τ, ξ_x, ξ_y, η_x, η_y, k_1, k_2. In order to make the results of the stability analysis independent both of a specific curvilinear grid and of the time step τ it is extremely important to express the entries of the amplification matrix G in the terms of the non-dimensional similarity parameters $\kappa_1, \ldots, \kappa_6$ defined by equations (33). This can be conveniently done with *Mathematica* by using the transformation rules. Let, for example, `cp1` $= c\tau\xi_x$. Then we can introduce the notation `cp1` into the entry `gg[[1,1]]` of the matrix G with the aid of transformation rule

```
g[[1,1]]=g[[1,1]]/.c->cp1/(dt ksix)
```

The remaining parameters $\kappa_2, \ldots, \kappa_6$ are introduced in the entries of G in a similar way. The first row of G thus obtained is then stored in the file `row1.m`. After that we compute in a similar way the next rows of G. Such a strategy saves a lot of computer memory, so that only about 2 Mb are needed to compute one row of G.

Once we have computed all the rows of the matrix G, we can assemble them into a 4×4 matrix.

It turns out that the matrix G obtained in this way still depends explicitly on the time step τ. More precisely, G has the following structure:

$$G = \begin{pmatrix} f_{11} & f_{12} \cdot q & f_{13} \cdot q & f_{14} \cdot q^2 \\ \frac{f_{21}}{q} & f_{22} \cdot 1 & f_{23} \cdot 1 & f_{24} \cdot q \\ \frac{f_{31}}{q} & f_{32} \cdot 1 & f_{33} \cdot 1 & f_{34} \cdot q \\ \frac{f_{41}}{q^2} & \frac{f_{42}}{q} & \frac{f_{43}}{q} & f_{44} \cdot 1 \end{pmatrix},$$

where $q = \tau\xi_x$ and the f_{ij}, $i, j = 1, \ldots, 4$ are functions of $\vec{\kappa}$ only. Similarly to Ganzha and Vorozhtsov (1996a) it is easy to show that there exists a one-parameter family of diagonal matrices

$$Q = \mathrm{diag}(q^\alpha, q^{\alpha+1}, q^{\alpha+1}, q^{\alpha+2})$$

such that the matrix

$$G_0 = QGQ^{-1} \tag{6}$$

does not depend on q. A simple choice for Q is obtained at $\alpha = 0$:

$$Q = \mathrm{diag}(1, q, q, q^2).$$

The symbolic computation of G_0 in accordance with (6) can conveniently be implemented with *Mathematica*:

```
G0 = DiagonalMatrix[{1,q, q, q^{2}]*G*Inverse[DiagonalMatrix[{1,q, q,
      q^{2}}]]
```

As a result we obtain a matrix G_0 with non-dimensional entries, which can be obtained formally from the matrix G by setting $q = 1$. In what follows we will omit the subscript 0 and write G instead of G_0 for brevity.

In this way we have obtained the amplification matrix on a curvilinear grid for Jameson's scheme (20), which takes 1898 lines of text, with 65 symbols in each line on the average.

6.2. NUMERICAL STAGES

In the method of Ganzha *et al.* (1992) the Newton identities were used to compute numerical values of the coefficients $c_j(\vec{\kappa}, \vec{k})$ of the characteristic equation (1). We first implemented this approach in a *Mathematica* program. But it turned out that a numerical computation of the coordinates of several points of the stability region boundary with this method requires many dozens of hours of CPU time of a powerful workstation. The main reason for these very large expenses of CPU time is that in the method of Newton identities the values of $c_j(\vec{\kappa}, \vec{k})$ should be computed at any point of a grid in the two-dimensional square region (28).

Therefore, it is very desirable to compute the coefficients $c_j(\vec{\kappa}, \vec{k})$ in a symbolic form only once. At the subsequent numerical stages, the numerical values of $\vec{\kappa}$ and $\vec{k}$ will be substituted into the expressions for $c_j(\vec{\kappa}, \vec{k})$. However, the internal representation of the entries of the amplification matrix G for scheme (20) on a curvilinear grid takes more than 8 MB. The intermediate expressions arising in the process of symbolic computation of the

coefficients $c_j(\vec{\kappa}, \vec{k})$ would require at least an order of magnitude larger memory space as this follows from the estimates presented in Ganzha and Vorozhtsov (1996a). Therefore, the symbolic computation of the coefficients $c_j(\vec{\kappa}, \vec{k})$ corresponding to a general matrix G is out of question.

We have implemented another strategy. Let us denote the entries of the amplification matrix G by g_{jk}, $j, k = 1, \ldots, 4$. The expression for the entry g_{11} takes 134 lines of text and is represented by a sum of many monomials. Each monomial is a rational function of $\vec{\kappa}, \vec{k}$ and γ (the constant γ enters the ideal gas equation of state (3)). Let us consider one of these monomials:

```
-(I/4*cp1^2*cp3*cp4^2*cp5*g*sb^3)/(-1 + g),
```

where `cp1` $= \kappa_1$, `cp3` $= \kappa_3$, `cp4` $= \kappa_4$, `cp5` $= \kappa_5$, `g` $= \gamma$, `sb` $= \sin k_2$. The length of this monomial can be reduced considerably by specifying the value of γ: $\gamma = \frac{7}{5}$ for air. We have used this value of γ in all our runs. As result the above monomial takes a shorter form

```
(-7*I)/8*cp1^2*cp3*cp4^2*cp5*sb^3
```

Further simplification of the entries of G can be performed by considering certain sections in the Euclidean space E^6 of $(\kappa_1, \ldots, \kappa_6)$ points. For example, if we cut the stability region $D \subset E^6$ by a hyperplane $\kappa_3 = 0$, the above monomial along with many other monomials vanish, which leads to a drastical simplification of G. In this connection we performed the numerical runs in the following sequence.

> At first a specific section in E^6 was specified. For example, to consider a particular case of a rectangular uniform spatial mesh we must specify $\kappa_4 = 0$, $\kappa_6 = 0$ (see Section 5).
>
> The corresponding matrix G is computed in symbolic form. The specification of the value $\gamma = \frac{7}{5}$ reduces the length of the expression for G from 1898 lines to 680 lines.
>
> The coefficients $c_j(\vec{\kappa}, \vec{k})$ of the characteristic equation (1) are computed in symbolic form by using the *Mathematica* function
>
> ```
> poly4=Det[z IdentityMatrix[4]-G]
> ```

This strategy has enabled us to reduce the needed CPU time at the numerical stage of our symbolic–numeric method by a factor of about 7 in comparison with the method of Ganzha *et al.* (1992).

Let us denote by Π_h a rectangular grid which covers the square region (28). In accordance with (35) the eigenvalues of G should be computed in all nodes of Π_h. Therefore, the next possibility of reducing the CPU time at a numerical stage is to reduce the size of the region Π. For example, in the case of the Jameson's scheme (20) on a rectangular uniform grid it turns out that the coefficients $c_j(\vec{\kappa}, \vec{k})$ involve only the even powers of $\sin k_1$ and $\sin k_2$ and do not contain $\cos k_1$ and $\cos k_2$. In this case we can consider instead of the periodicity region (28) a much smaller region

$$\Pi_1 = \left\{ (k_1, k_2) | 0 \le k_1 \le \frac{\pi}{2}, \quad l = 1, 2 \right\} \tag{7}$$

because the function $\sin^2 k_1$ takes on all its admissible values $0 \le \sin^2 k_1 \le 1$ in the interval $0 \le k_1 \le \frac{\pi}{2}$. Since the ratio of areas of the regions Π and Π_1 is equal to 16, we have expected that the needed CPU time at numerical stage would reduce also by a factor of 16. However, the actual CPU time proved to be only by a factor of 4 smaller than in the case of using the region Π (28). We will present an explanation of this effect in the following after we describe Steps 1 and 2 of our numerical algorithm.

Let us now present a mathematical formulation of the problem of searching for the boundary points of the necessary stability region of a difference scheme whose characteristic equation has the form (1). Let $\lambda_j(\vec{\kappa}, \vec{k})$ be the eigenvalues of G. Following Ganzha and Vorozhtsov (1996a) we introduce the functions

$$g_j(\vec{\kappa}, \vec{k}) = 1 - |\lambda_j(\vec{\kappa}, \vec{k})|^2, \qquad j = 1, \ldots, 4. \tag{8}$$

Then the von Neumann necessary stability criterion reduces to the system of inequalities

$$g_j(\vec{\kappa}, \vec{k}) =\geq 0, \qquad j = 1, \ldots, 4.$$

Let us assume that the left-hand sides of these inequalities are periodic functions in k_1 and k_2 with periods T_1 and T_2, respectively, and let

$$\Pi_2 = \{(k_1, k_2) | 0 \le k_m \le T_m, \quad m = 1, 2\}. \tag{9}$$

Let us choose the quantity κ_1 as a parameter and suppose that the functions

$$\kappa_1 = \kappa_1, \quad \kappa_j = \varphi_j(\kappa_1), \qquad j = 2, \ldots, 6 \tag{10}$$

determine parametrically some smooth curve $\mathcal{L}$ in E^6 that intersects the boundary Γ of the necessary stability region D of a difference scheme under study. Let

$$\tilde{g}_k(\kappa_1, \vec{k}) = g_k(\kappa_1, \varphi_2(\kappa_1), \ldots, \varphi_6(\kappa_1), \vec{k}), \qquad k = 1, \ldots, 6. \tag{11}$$

We now form the following binary function $f(\kappa_1)$:

$$f(\kappa_1) = \begin{cases} 1, & \tilde{g}_m(\kappa_1, \vec{k}) \ge 0 \forall \vec{k} \in \Pi_2, \qquad m = 1, \ldots, 4 \\ -1 & \text{otherwise.} \end{cases} \tag{12}$$

Then the value $\kappa_1^0 \in \Gamma$, if the function $f(\kappa_1)$ has a jump at point $\kappa_1 = \kappa_1^0$ and changes its sign.

We have determined $\kappa_1^0 \in \Gamma$ numerically as follows. Let $\kappa_1^*(k_2)$ be such a value at fixed $k_2 \in [0, T_2]$ that $\tilde{g}_k(\kappa_1^*(k_2), k_1, k_2) \ge 0$, but $\tilde{g}_k(\kappa_1^*(k_2) + \delta, k_1, k_2) < 0$ at least for some $k, 1 \le k \le 4$, and for some $k_1 \in [0, T_1]$, and any small positive $\delta > 0$. We now introduce the set

$$A(k_1, k_2^0) = \mathrm{Arg} \min_{0 \le k_1 \le T_1} \kappa_1^*(k_1, k_2^0). \tag{13}$$

This set typically contains several points. Example: the function $\kappa_1^* = \sin 4k_2$ has two equal minima at $k_2^{(1)} = \frac{3\pi}{8}$ and $k_2^{(2)} = \frac{5\pi}{8}$ in the interval $0 \le k_2 \le \pi$. The point $\kappa_1^0 \in \Gamma$ is then determined as

$$\kappa_1^0 = \min_{k_2 \in [0, T_2]} \{ \min_{k_1 \in A(k_1, k_2)} \kappa_1^*(k_1, k_2) \}. \tag{14}$$

This means that we initially determine $\kappa_1^*(k_1, k_2)$ at a fixed value of $k_2 \in [0, T_2]$. In practical computations the lines $k_2 = \text{const}$ were the lines of one family of grid lines of a rectangular region Π_2 (9).

The accuracy of the computation of value $\kappa_1^0 \in \Gamma$ depends on the accuracy with which

the zeroes $\lambda_j(\vec{\kappa}, \vec{k})$ of the characteristic equation (1) are computed. As was shown in Ganzha and Vorozhtsov (1996a), the arithmetic of floating-point machine numbers becomes insufficient in the case of complex entries of G. The reason for this is a rapid accumulation of roundoff errors. Two methods were previously used in Ganzha and Vorozhtsov (1996a) to reduce the effect of roundoff errors:

(i) the balancing of matrices and the arithmetic of stored orders;
(ii) the use of double precision machine arithmetic.

However, it was stressed in Ganzha and Vorozhtsov (1996a) that the double precision arithmetic can also lead to large errors when sufficiently complex difference schemes are to be analysed.

The system *Mathematica* contains a number of numerical functions to perform the numerical computations with high accuracy: these are the functions

```
N[expr,n], SetPrecision[expr,n], SetAccuracy[expr,n].
```

It was shown in Ganzha *et al.* (1996) that these numerical functions yield somewhat lower accuracy than that specified by the argument n for certain ranges of n. In this connection it was proposed in Ganzha *et al.* (1996) to use the arithmetic of rational numbers. In this method, the trigonometric functions `Sin[x], Cos[x]` are approximated by rational numbers with the aid of the built-in *Mathematica* function `Rationalize[expr,eps]`:

```
si[xi_, n_]:= si[xi,n] = Rationalize[N[Sin[xi], n+1], 10^-(n+1)];
co[xi_, n_]:= co[xi,n] = Rationalize[N[Cos[xi], n+1], 10^-(n+1)]
```

The second argument `eps` of the function `Rationalize[exp,eps]` denotes the maximum difference of the rational value from the exact value rather than the number of digits. If one enters `eps=0`, then *Mathematica* tries to attain the best possible fraction.

The values of $\kappa_1, \ldots, \kappa_6$ in the coefficients $c_k(\vec{\kappa}, \vec{k})$ were also specified as rational numbers. As a result, all the coefficients $c_k(\vec{\kappa}, \vec{k})$ were rational numbers. At given rational values of the coefficients c_k, the characteristic equation (1) was solved with the aid of the *Mathematica* function `Solve[]`.

Similarly to Ganzha and Vorozhtsov (1996a) we have used the bisection method to find the value $\kappa_1^*(k_2)$ at a fixed value of k_2. In accordance with (12) the value of $\kappa_1^*(k_2)$ is determined at various $k_1 \in [0, T_1]$ as the maximum value for which $f(\kappa_1) = 1$ holds. Since the above bisection process should be applied for any value of k_1 in $[0, T_1]$, much computer time can be required to find the value $\kappa_1^*(k_2)$. In this connection we have implemented a two-step process for determining the value κ_1^0 in accordance with (14).

Step 6.1. A rough computation of κ_1^0. Let us assume that we want to compute the value of κ_1^0 at first with the accuracy 10^{-n2}, where $1 \leq n2 < exact$, *exact* is the user-specified desired number of correct digits in the mantissa of the result. Let us call a partition of the interval $[0, T_1]$ into 2^n equal subintervals the nth partition, $n = 0, 1, 2, \ldots$. At the zeroth partition we take the interval $[0, T_1]$ itself. At the first partition we already have two intervals $[0, T_1/2]$ and $[T_1/2, T]$, etc. Let us denote by $\kappa_1^{(n)}$ the minimum value of κ_1 at the nth partition, which is obtained at fixed values of k_1 in the nodes of the nth

partition, and let $\bar{\kappa}^{(n)}$ be the minimum value of κ_1 over the set of remaining nodes in the interval $[0, T_1]$. If $|\kappa_1^{(n)} - \bar{\kappa}_1^{(n)}| < 10^{-n2}$, the process of the further partition of the interval $[0, T_1]$ is terminated. It is easy to find that the nth partition adds only 2^{n-1} new nodes to the $(n-1)$th partition. This has enabled us to store the values of $\kappa_1^*(k_1)$ at the foregoing partition in the nodes of a cruder mesh on the interval $0 \le k_1 \le T_1$. This saves a lot of computing time.

The value of $\kappa_1^*(k_2)$ in the interval $0 \le k_2 \le T_2$ is determined in a similar way by using the partition of this interval into 2^n equal subintervals. Let us denote by `[k2min, k2max]` $\subset [0, T_2]$ the interval in which the value $\kappa_1^*(k_2)$ is located.

Step 6.2. Accurate computation of κ_1^0. At this step we compute the value of κ_1^0 in accordance with (14) with the high accuracy $10^{-\text{exact}}$. Here we use the bisection process in the interval `[k2min, k2max]` found at Step 1. As a result we find a point $(\kappa_1^0, \varphi_2(\kappa_1^0), \ldots, \varphi_6(\kappa_1^0))$ of the necessary stability region boundary for a given difference scheme. The presented method of using the partitions of the intervals $[0, T_1]$ and $[0, T_2]$ into 2^n subintervals has a resemblance to the fast Fourier transform (Cooley and Tukey, 1965) in that the FFT algorithm uses the same type of partition.

Thus, we have presented all the details of the implementation of our numerical algorithm. Therefore, we can now estimate the effect of the size of the periodicity region (7) or (28) on the computing time needed to achieve the specified absolute error ε. Let T_1 be the size of the periodicity interval along the k_1 axis. It is clear that in order to achieve the accuracy ε one must perform n_1 partitions of the interval $[0, T_1]$, where n_1 is determined from the relation

$$\varepsilon = O\left(\frac{T_1}{2^{n_1}}\right). \tag{15}$$

Let us now take another periodicity interval $T_1' = T_1/4$. To achieve the same accuracy ε in this interval we need n_2 partitions of the interval $[0, T_1']$, where n_2 is found from the relation $\varepsilon = O(T_1'/(2^{n_2}))$. Taking (15) into account we then obtain that $n_2 \approx n_1 - 2$.

We now take into account the fact that 2^{n_1-1} new nodes are added to the (n_1-1)th partition at the n_1th partition. It is clear that the needed CPU time is proportional to the total number of the nodes in each of which we have to compute the characteristic polynomial zeros. This total number of nodes is obviously the geometric progression

$$1 + 2 + 2^2 + \cdots + 2^{n_1-1} = \frac{2^{n_1} - 1}{2 - 1} = 2^{n_1} - 1.$$

When using a smaller interval $T_1' = T_1/4$ the total number of nodes will be $2^{n_1-2} - 1$, respectively. Hence in the case of using the periodicity interval $[0, T_1]$ we will need the CPU time, which will be by a factor of

$$\frac{2^{n_1} - 1}{2^{n_1-2} - 1} = 4 + \frac{3}{2^{n_1-2} - 1} \tag{16}$$

larger than in the case of using the interval having the length $T_1/4$. In particular, at $\varepsilon = 10^{-2}$ and $T_1 = 2\pi$ we obtain from (15) that $n_1 \approx \log_2(2\pi/\varepsilon) \approx 9$, and then we obtain from (16) the factor $4 + \frac{3}{2^7-1} \approx 4$, which agrees very well with the above-presented actual estimate of the CPU time reduction when passing from $T_1 = 2\pi$ to $T_1' = \pi/2$.

In Table 2 we show how we compute each specific expression *expr* at the numerical

Table 2. The modes of computation of various quantities at the numerical stages.

$\sin k_m$, $\cos k_m$, $m = 1, 2$	$\kappa_1, \ldots, \kappa_M$	The $c_j(\kappa, \vec{k})$ in (1)	The roots of (1)	The value of κ on the stability region boundary
FLOPS	SC	SC	SC	SC

Table 3. Values of κ_1 at point $\kappa_2 = \kappa_3 = 0$ of the stability region boundary of scheme (20) at different values of κ_5.

$\kappa_1 \setminus \kappa_5$	$\frac{1}{2}$	1	2	3
$\kappa_{1\text{num}}$	$\frac{1831}{1024}$ $\approx 1.788\,086$	$\frac{5793}{4096}$ $\approx 1.414\,307$	$\frac{7325}{8192}$ $\approx 0.894\,165$	$\frac{971}{1536}$ $\approx 0.632\,162$
$\kappa_{1\text{ex}}$	1.788 854	1.414 214	0.894 427	0.632 456
$\delta\kappa_1$	0.000 768	0.000 093	0.000 262	0.000 294

stages of our algorithm. We use the following notations for the modes of computation of an expression: FLOPS is the computation with the aid of the arithmetic of the machine floating-point numbers; SC is the symbolic computation of an expression.

It follows from Table 2 that the symbolic mode of computation plays a predominant role at the numeric stages of our algorithm.

For the validation of the proposed numerical method we have considered the stability of scheme (20) in the particular case of a uniform rectangular spatial grid. Then we must take the values $\kappa_4 = 0$ and $\kappa_6 = 0$ in the coefficients of the characteristic equation. In order to be able to use the analytic formula (39) we have taken the values $\kappa_2 = \kappa_3 = 0$ (see the top of the pyramid in Figure 3). We have performed a number of runs with `n2=1` and $exact = 2$ at different values of the cell aspect ratio κ_5. We have used the following notations: $\kappa_{1\text{num}}$ is the value of κ_1 on the boundary of the stability region obtained by the above presented numerical method;

$$\kappa_{1\text{ex}} = \frac{2}{\sqrt{1 + \kappa_5^2}};$$

and $\delta\kappa_1$ is the absolute error, $\delta\kappa_1 = |\kappa_{1\text{ex}} - \kappa_{1\text{num}}|$. It can be seen from Table 3 that the absolute error $\delta\kappa_1$ is smaller by an order of magnitude than the user-specified accuracy 10^{-2}. The proposed numerical algorithm thus computes the coordinates of points of the stability region boundary with a guaranteed accuracy. This is explained by the fact that the arithmetic operations on rational numbers are exact in *Mathematica*. The *Mathematica* function `Solve[]` also finds the exact analytic expressions for the zeros of a fourth-degree polynomial with rational coefficients.

In Table 4 we present the numerical results obtained for the case of a curvilinear spatial grid. The notations $\kappa_{1\text{num}}$ and $\delta\kappa_1$ have the same meaning as in Table 3;

$$\kappa_{1\text{ex}} = \frac{2}{\sqrt{(1 + |\kappa_4|)^2 + \kappa_5^2}}.$$

The numerical data of Table 4 also confirm the correctness of the analytical formula (35). In the case where $\kappa_2 = \kappa_3 = \kappa_6 = 0, \kappa_4 \neq 0$, $\kappa_5 \neq 0$ the coefficients of the characteristic equation involve both even and odd powers of $\sin k_1$ and $\sin k_2$. Therefore, our *Mathe-*

Table 4. Values of κ_1 at $\kappa_2 = \kappa_3 = \kappa_6 = 0$ on the necessary stability region boundary of scheme (20) at different values of κ_4 and κ_5.

$\kappa_1 \setminus \kappa_4, \kappa_5$	$\kappa_4 = 1, \kappa_5 = 2$	$\kappa_4 = 2, \kappa_5 = 1$	$\kappa_4 = 3, \kappa_5 = 3$	$\kappa_4 = -\frac{1}{2}, \kappa_5 = -\frac{1}{3}$
$\kappa_{1\text{num}}$	$\frac{915}{1024}$ $\approx 0.893\,555$	$\frac{971}{1536}$ $\approx 0.632\,162$	$\frac{409}{1024}$ $\approx 0.399\,414$	$\frac{5331}{4096}$ $\approx 1.301\,514$
$\kappa_{1\text{ex}}$	0.894 427	0.632 456	0.400 000	1.301 583
$\delta\kappa_1$	0.000 873	0.000 294	0.000 586	0.000 069

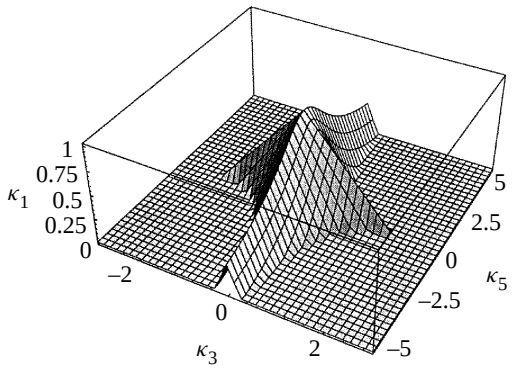

Figure 4. The necessary stability region of Jameson's scheme on the curvilinear grid.

matica program has computed the periods T_1 and T_2 in k_1 and k_2 and $T_1 = T_2 = 2\pi$. In the last variant with $\kappa_4 = -\frac{1}{2}, \kappa_5 = -\frac{1}{3}$ the value of $\kappa_{1\text{num}}$ was found at $k_1 = \frac{3\pi}{2}$, $k_2 = \frac{\pi}{2}$.

In Figure 4 we show the stability region of scheme (20) in the section $\kappa_2 = 0, \kappa_4 = 1, \kappa_6 = 1$. This case also corresponds to the curvilinear grid.

7. Results

The numerical results presented in Tables 3 and 4 confirm the validity of the stability condition (35) for scheme (20). Therefore, inequality (35) can be used for the computation of a local time step $\tau_{j,k}^{n+1}$ in each node (x_{jk}, y_{jk}) of a curvilinear grid. For this purpose the local numerical values $c_{jk}^n, u_{jk}^n, u_{jk}^n, (\xi_x)_{jk}, (\xi_y)_{jk}, (\eta_x)_{jk}, (\eta_y)_{jk}$ are substituted in (36). As a result the local value $\tau_{j,k}^{n+1}$ can be computed by formula (36) with $C = 2$.

For the numerical solution of transonic flow problems by the pseudo-unsteady method the initial conditions for the Euler equations (1)–(3) are usually specified as a homogeneous free stream with a given freestream Mach number M_∞ and a given angle of attack α.

We consider in what follows the effect of these freestream conditions on local time steps $\tau_{jk}^{n+1}, n = 0$, which are allowed by the stability condition (36) of difference scheme (20). These conditions should be substituted into the difference equations to obtain the difference solution $\vec{u}^1$ at the first time level $t = \tau_{jk}^1$. It is known that the transients are large at the first several time steps of the pseudo-unsteady method, so that care must be taken in choosing the time steps from the stability requirement (Roache, 1976). At a certain

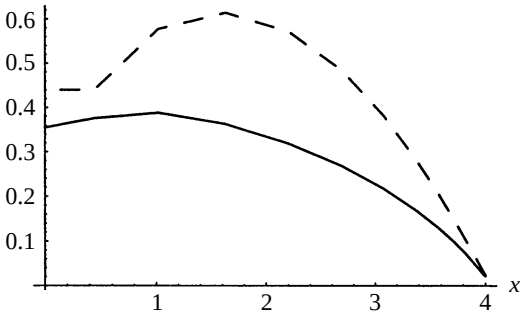

Figure 5. – – –, The graph of $(x_\eta)_{j1,k}$; —, $\tau = \tau_{\max}(x, 0)$ as computed by (36) at $\theta = 1$.

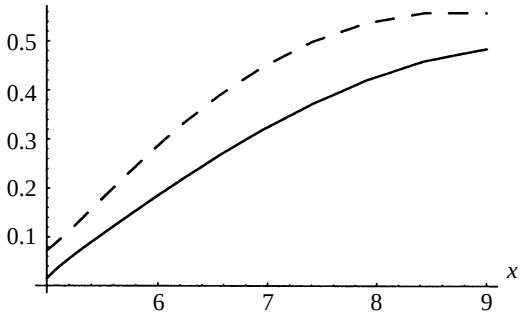

Figure 6. – – –, The graph of $(x_\eta)_{j1,k}$; —, $\tau = \tau_{\max}(x, 0)$ as computed from (36); angle of attack $\alpha = 0$. O-grid.

integer value $j = j_1 > 0$ the grid nodes of the C-grid around the airfoil NACA0012 are located on a segment of the x-axis between the coordinate origin $(0, 0)$ and the airfoil leading edge; in our case this is the interval $0 \le x \le 4$ (see also Figure 1a). Let us compute the numerical values of the derivative $(x_\eta)_{j_1,k}$ as

$$(x_\eta)_{j_1,k} = x_{j_1,k+1} - x_{j_1,k}, \quad k = 1, \ldots, K-1; \quad (x_\eta)_{j_1,K} = x_{j_1,K} - x_{j_1,K-1}.$$

In Figure 5 we show the graphs of $(x_\eta)_{j_1,k}$ and the graph of $\tau_{\max}$. This is the maximum value of the local time step τ_{jk}^{n+1} obtained from the right-hand side of (36) at $\theta = 1$ for the zero angle of attack α. It can be seen from Figure 5 that the local size of the maximum time step is determined to a large extent by the local values of the metric derivative $(x_\eta)_{j,k}$. This is not surprising, because the derivative $x_\xi \approx 0$ in the considered interval of the x-axis (cf. Figure 1a): the grid lines are nearly vertical; it is obvious that $(y_\eta)_{j_1,k} = 0$ on the axis $y = 0$. The same qualitative behaviour of $\tau_{\max}(x, 0)$ takes place also in the case of the O-grid shown in Figure 1c, (see Figure 6).

On the basis of (36) we can compute the maximum value $\tau_{\max}$ at each grid point (x_{jk}, y_{jk}) by taking the value $\theta = 1$. In this way we can obtain a surface in the three-dimensional Euclidean space of (x, y, τ)-points. We will call this surface the τ-surface in the following.

In Figure 7 we show the τ-surface for the case of C-grid and zero angle of attack α. All the τ-surfaces were generated with the aid of the *Mathematica* function `ParametricPlot3D[]`. The functions $x = x(s_1, s_2), y = y(s_1, s_2), \tau = \tau(s_1, s_2)$ spec-

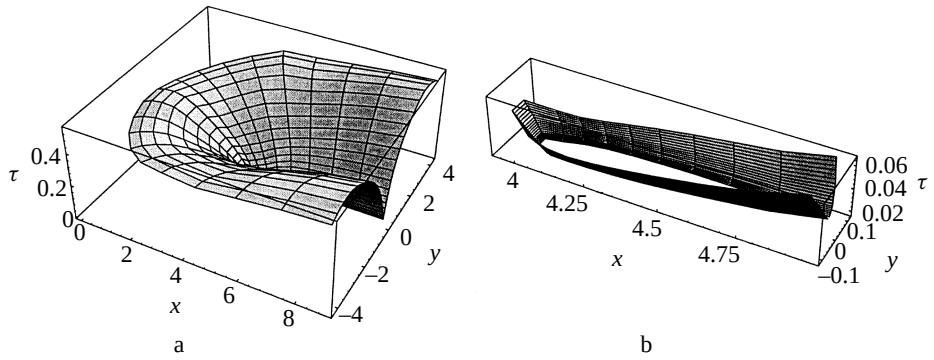

Figure 7. τ-surface in the case of the C-grid of Figure 1a. a, The complete τ-surface; b, partial view of τ-surface in the vicinity of airfoil surface.

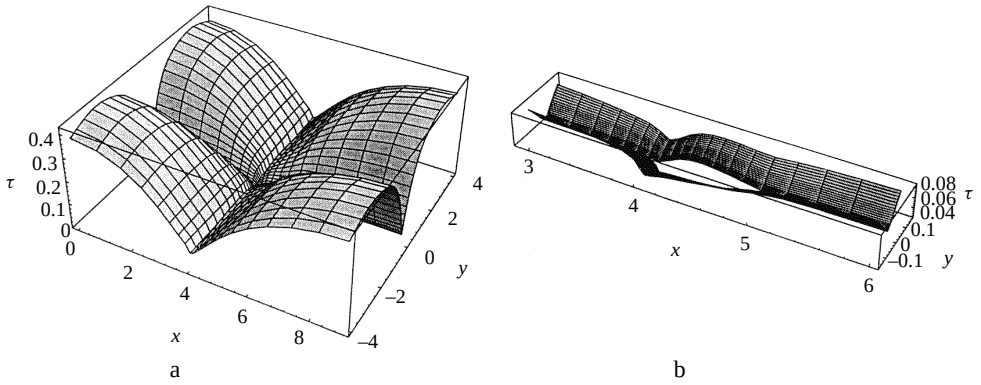

Figure 8. τ-surface in the case of the H-grid of Figure 1b. a, The complete τ-surface; b, partial view of τ-surface in the vicinity of airfoil surface.

ifying parametrically the τ-surface were determined with the aid of linear transfinite interpolation (Knupp and Steinberg, 1994). In Figures 8 and 9 we show the τ-surface for the case of H-grid and O-grid, respectively, and the zero incidence α. It can be seen from Figures 7–9 that the τ-surfaces have qualitatively different shapes for different grid topologies. It is also seen that the height of the τ-surfaces is smaller in the regions with small local sizes of the mesh cells. It can be seen from Figures 5–9 that the values of the time step τ are determined to a large extent by the local values of the metric derivatives $x_\xi, x_\eta, y_\xi, y_\eta$.

The requirement of accuracy of the numerical solution dictates that the sizes of grid cells be small near the airfoil surface. But this leads to very small local values of the maximum time step $\tau_{\max}$ its size near the airfoil turns out to be smaller by an order of magnitude than the value of $\tau_{\max}$ in the far field.

Thus, the requirement of uniform stability robustness of an explicit difference scheme on a curvilinear grid contradicts the accuracy requirement. These big differences in the local values of τ can be reduced by increasing the grid uniformity throughout the spatial region around the airfoil. With regard to the accuracy demands, this improved curvilinear

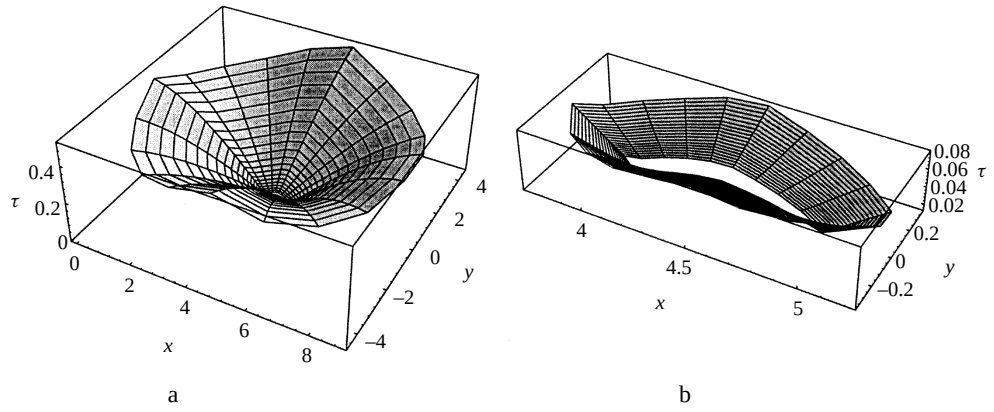

Figure 9. τ-surface in the case of the O-grid of Figure 1c. a, The complete τ-surface; b, partial view of τ-surface in the vicinity of airfoil surface.

grid uniformity can be achieved at the expense of a considerable increase of the number of grid nodes in each of the ξ- or η-directions.

Another way to make the values $\tau_{\max}$ grid-independent is to use the absolutely stable difference schemes on a curvilinear grid.

We have also considered the effect of the angle of attack α in the initial freestream condition on the local behavior of $\tau_{\max}$. For this purpose we have taken the range $0 \leq \alpha \leq 10°$. In Figure 10 we show the graphs of $\tau_{\max}$ along the same segment of the x-axis as in Figure 6.

We can see from Figure 10 that the maximum value of $\tau_{\max}$ in the interval $5 \leq x \leq 9$ slightly reduces as α increases. Thus, the geometric effects of curvilinear grid on local stability become less significant with increasing angle of attack α.

At $n > 0$ formula (36) can be used simply by substituting the computed grid values of u^n_{jk}, v^n_{jk} and c^n_{jk} into the right-hand side of this formula.

8. *Mathematica* vs. REDUCE

In this section we present a comparison of the systems *Mathematica* and REDUCE from the viewpoint of their applicability for the stability analysis of finite-difference and finite-volume methods for the numerical integration of the Euler or Navier–Stokes equations of compressible fluids in cases of two or three spatial variables. We have indeed discussed in the foregoing sections the advantages and shortcomings of the both systems when they are applied for the stability analysis. We want to summarize here these discussions and draw the final conclusion. For the convenience of the further comparisons we itemize various aspects or criteria in accordance with which we compare the both CASs.

(1) When the REDUCE is used for the stability analysis of the finite-difference or finite-volume schemes it is necessary to perform the numerical stages of the analysis by going over to FORTRAN, C or some other language of the numerical computations. This is not convenient for the user from the viewpoint of an efficient execution of the symbolic–numeric computations and from the viewpoint of the reduction of the needed computer storage.

On the other hand, one can initiate the floating-point numerical computations in *Math-*

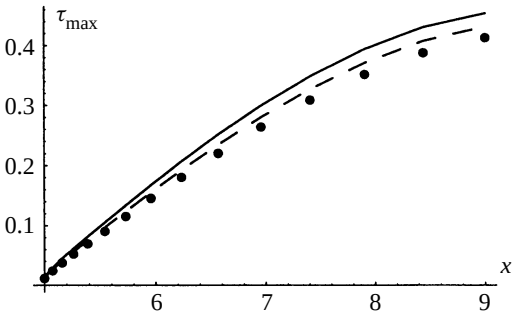

Figure 10. The graphs of $\tau_{\max}$: —, $\alpha = 0$; – – –, $\alpha = 5°$; $\cdots$, $\alpha = 10°$. The O-grid.

ematica at any point of the program. After that one can resume the symbolic computations in the same program. In our program described above we use just this alternation of the symbolic and numeric computations within the framework of the same *Mathematica* program: at some section of the Euclidean space of κ points the characteristic polynomial is computed in symbolic form, then the values of $\sin k_2$ and $\cos k_2$ are computed for a given value of the wavenumber k_2 and their rational approximations are substituted in the characteristic polynomial. After that a shorter polynomial is again computed in symbolic form. Then a sequence of the value of the wavenumber k_1 is substituted in this polynomial for the purpose of obtaining the set in the right-hand side of equation (13) with the aid of numerical computations. Then in accordance with (14) a new value of the wavenumber k_2 is specified, and the above sequence of symbolic and numeric computations is repeated until we obtain the set in the right-hand side of (14). Such an approach often leads to a reduction of the amount of computations from the viewpoint of the needed computer storage.

The user needs only to input the difference equations to be analyzed, the form of the non-dimensional variables $\kappa_1, \ldots, \kappa_M$ and the region of their variation. Then the user only waits for the needed result, the stability region. Such an integrity of the analysis program is usually attractive for industrial engineers. We have presented examples of such *Mathematica* programs for the stability analysis of scalar two-level difference schemes in Ganzha and Vorozhtsov (1996b).

(2) A feature of the Fourier method is the fact that the coefficients of the characteristic equation (1) depend on the trigonometric functions $\cos k_m$, $\sin k_m$, $m = 1, 2$ (see, e.g. equation (5)). To avoid the accumulation of roundoff errors at the numerical stages we have performed the computations in the *Mathematica* program with the use of the arithmetic of the rational numbers. However, the functions $\cos x$ and $\sin x$ have rational values only for several values of the argument x in the periodicity interval. Hence a function is needed which enables one to convert the values of $\sin x$ and $\cos x$ as the floating-point numbers into the rational numbers with a user-specified accuracy. Such a function `Rationalize[]` is available in *Mathematica*. It makes no sense to develop such a function within the framework of REDUCE, because the arithmetic of the machine floating-point numbers is not included in REDUCE.

(3) The advantage of REDUCE 3.6 is a possibility of declaring the non-commutative operators, which enables one to correctly calculate the algebraic expressions involving these operators. In this respect, *Mathematica* gives way to REDUCE. We had to perform a long and painful search for the solution of the problem of non-commutative mul-

Table 5. *Mathematica* vs. REDUCE.

Questions	*Mathematica*	REDUCE
Is it possible to perform the symbolic and numerical stages within the same program?	Yes	No
Is it possible to convert the machine floating-point numbers into the rational numbers with a user-specified accuracy?	Yes	No
Is it possible to declare the variables or functions as non-commutative?	No	Yes
Are there the convenient built-in computer graphics?	Yes	No

tiplication of arbitrary variables within the framework of *Mathematica 2.2*. There is no progress in this respect in *Mathematica 3.0*. In our opinion, the principal solution of the problem would be the introduction of the declaration `non-commutative`, which could be applicable not only to the functions but also to the `variables`.

However, the other users of the system *Mathematica* have the right to require the introduction of the declarations of their other abstract operations. Therefore, a more general approach to the solution of the problem of non-commutative multiplication within the framework of computer algebra systems consists of the inclusion in the system of certain abstract operations the algebraic rules of which are fully made explicit by the user. However, it is necessary to ensure here the CAS immunity to the intervention of "foreign" operations. Otherwise these operations will have the effect of virus.

(4) The specifics of the stability analysis of difference schemes for fluid dynamics problems consist of the fact that the analysis result, the stability region, has a clear geometric representation. It enables the user to rapidly compare different numerical methods from the viewpoint of their stability properties.

Mathematica has a powerful set of the built-in functions of computer graphics. These functions are very convenient to use. REDUCE 3.6 possesses in this respect much more modest capabilities (see Hearn, 1995).

(5) As was already noted in Section 3, the coordinates of the curvilinear grid nodes are machine floating-point numbers. The specific values of the time steps, which are allowed by the stability, depend not only on the number of grid nodes but also on the grid topology, see Figures 5–10. The CAS *Mathematica* enables one to obtain the information about these maximal steps, since one can use in the *Mathematica* program the arithmetic of the machine floating-point numbers.

We summarize the above discussion in Table 5.

It follows from Table 5 that the CAS *Mathematica* is undoubtedly better suited to the problems of symbolic–numeric stability investigation than the CAS REDUCE 3.6.

9. Summary

The main purpose of the above research was the extension of our previous stability analysis method (see Ganzha and Vorozhtsov, 1996a) to curvilinear grids on the basis of implementation within *Mathematica*. The presented symbolic–numeric method is efficient both in terms of the requirements for memory/speed of computer (they are modest) and

in terms of the numerical accuracy (this accuracy can be specified by the user, because our method does not introduce any roundoff errors).

In what follows we outline some further extensions of the above stability analysis method. The above method gives exact results on the stability regions in cases of parallelogram grids. These results become approximate, but are still applicable in practice when the curvilinear grid deviates from the parallelogram grid in some spatial subregions. The applicability of formula (36) in the case of a general curvilinear grid can be achieved by taking a value of safety factor θ in the interval $0 < \theta < 1$. However, this can lead to underestimation of time step in comparison with the actual maximum time step allowed by stability on a general curvilinear grid. This leads to an increase in computer time expenses while solving a specific fluid flow problem. In this connection the next obvious step in the extension of the above method is the consideration of a general curvilinear grid. It is easy to show that this consideration would involve nine non-dimensional variables $\kappa_1, \ldots, \kappa_9$ instead of six variables (33). Thus, the complexity of computer-aided stability analysis increases in the case of a general curvilinear grid.

Since a successful numerical modeling of aerodynamics problems involving shock waves necessitates the introduction of the artificial dissipation terms in the numerical discretization schemes, the next step in the extension of the above presented symbolic–numeric method would be stability analysis of numerical methods with regard for the above extra terms.

The most difficult problem in the stability analysis of numerical methods of computational fluid dynamics (CFD) is the stability analysis of finite-difference or finite-volume schemes in the presence of artificial dissipators on three-dimensional curvilinear grids. Our estimates of algorithmic complexity, which we presented previously in Ganzha and Vorozhtsov (1996a) show that very powerful computers (supercomputers) are needed to solve this problem. The solution of this stability analysis problem is particularly important for turbulence modeling. The reliable stability predictions should help the researchers of turbulence to distinguish clearly between the onset of numerical instability and the onset of turbulence in a specific flow problem.

With regard for the extension of the above stability analysis method to three-dimensional fluid flow problems on curvilinear grids it would be interesting to perform the parallelization of our analysis method in the *Mathematica 3.0* environment for the purpose of the minimization of computer times needed both at symbolic and numeric stages of our method. This work is now in progress.

There is an interesting research theme still in the field of two-dimensional CFD problems: the stability investigation of finite-difference and finite-volume methods on unstructured grids. Such grids now become more and more popular in CFD.

The stability investigation of difference schemes is only one stage on the way of choosing and developing the computer codes for the numerical modeling of fluid dynamics processes. From the viewpoint of engineering applications, the development of the above computer codes implementing a specific numerical method is of larger significance. Our experience in the application of the CAS *Mathematica* for the numerical generation of curvilinear grids (see Section 3) as well as for the numerical solution of two-dimensional mathematical physics problems governed by scalar partial differential equations (Ganzha and Vorozhtsov, 1996b) points to the fact that this CAS can be applied successfully also for the development of a computer code for the computation of transonic fluid flow around airfoils or wings in the cases where the difference scheme is explicit (as the Runge–Kutta-type schemes).

References

Chakravarthy, S. R. (1983). Euler equations-implicit schemes and boundary conditions. *AIAA J.*, **21**, 699–706.

Cooley, J., Tukey, J. (1965). An algorithm for the machine calculation of complex Fourier series. *Math. Comput.*, **19**, 297–301.

Eiseman, P. R. (1979). A multi-surface method of coordinate generation. *J. Comput. Phys.*, **33**, 118–150.

Fletcher, C. A. J. (1996). *Computational Techniques for Fluid Dynamics,* 3rd edn*, volumes I and II*, Berlin, Springer.

Ganzha, V. G., Schaub, K. L., Vorozhtsov, E. V. (1996). On accurate computation of stability regions of difference schemes for fluid dynamics problems. In *Int. Conf. on the Methods of Aerophysical Research, September 2–6, 1996, Novosibirsk, Russia*, Proceedings Part I, pp. 96–101. Novosibirsk, Institute of Theoretical and Applied Mechanics.

Ganzha, V. G., Vorozhtsov, E. V. (1993). On the stability of Jameson schemes. *Bridging Mind and Model: Papers in Applied Mathematics*, Costa, P. J. ed., pp. 237–300. St. Paul, St. Thomas Technology Press.

Ganzha, V. G., Vorozhtsov, E. V. (1996a). *Computer-aided Analysis of Difference Schemes for Partial Differential Equations*, New York, Wiley.

Ganzha, V. G., Vorozhtsov, E. V. (1996b). *Numerical Solutions for Partial Differential Equations: Problem Solving Using Mathematica*, Boca Raton, FL, CRC Press.

Ganzha, V. G., Vorozhtsov, E. V., Boers, J., van Hulzen, J. A. (1994). Symbolic-numeric stability investigations of Jameson's schemes for the thin-layer Navier-Stokes equations. *Int. Symp. on Symbolic and Algebraic Computation*, von zur Gathen, J., Giesbrecht, M. eds., pp. 234–241. New York, ACM Press.

Ganzha, V. G., Vorozhtsov, E. V., van Hulzen, J. A. (1992). A new symbolic-numeric approach to stability analysis of difference schemes. *Int. Symp. on Symbolic and Algebraic Computation*, Wang, P. S. ed., pp. 9–15. New York, ACM Press.

Godunov, S. K., Ryabenkii, V. S. (1987). *Difference Schemes: An Introduction to theUnderlying Theory*, volume 19, Studies in Mathematics and its Applications. New York, Elsevier Science.

Gustafsson, B., Kreiss, H.-O, Sundström, A. (1972). Stability theory of difference approximations for mixed initial boundary value problems II. *Math. Comput.*, **26**, 649–686.

Hearn, A. C. (1995). *REDUCE User's Manual. Version 3.6*, Santa Monica, CA, *RAND Publication CP78 (Rev. 7/95)*, pp. 90407–92138.

Jameson, A., Schmidt, W. (1985). Some recent developments in numerical methods for transonic flows. *Comput. Methods Appl. Mech. Eng.*, **51**, 467–493.

Jameson, A., Schmidt, W., Turkel, E. (1981). Numerical solution of the Euler equations by finite-volume methods using Runge–Kutta time stepping schemes. AIAA Paper, 81–1259.

Knupp, P., Steinberg, S. (1994). *Fundamentals of Grid Generation*, Boca Raton, Ann Arbor, CRC Press.

Kovenya, V. M., Yanenko, N. N. (1981). *Splitting-up Method in Gas Dynamics Problems* (in Russian), Novosibirsk, Nauka.

Lax, P. D., Nirenberg, L. (1966). On stability for difference schemes: a sharp form of Gording's inequality. *Commun. Pure Appl. Math.*, **19**, 437–492.

LeVeque, R. J. (1992). *Numerical Methods for Conservation Laws*, Basel, Boston, Berlin, Birkhäuser Verlag.

Mavriplis, D. J. (1988). Multigrid solution of the 2-D Euler equations on unstructured triangular meshes. *AIAA J.*, **26**, 824–831.

Pike, J., Roe, P. L. (1985). Accelerated convergence of Jameson's finite-volume Euler scheme using van der Houwen integrators. *Comput. Fluids*, **13**, 223–236.

Richtmyer, R. D., Morton, K. W. (1967). *Difference Methods for Initial-value Problems*, New York, Wiley Interscience.

Roache, P. J. (1976). *Computational Fluid Dynamics*, Albuquerque, New Mexico, Hermosa.

Strikwerda, J. C. (1989). *Finite Difference Schemes and Partial Differential Equations*, Pacific Grove, CA, Wadsworth & Brooks/Cole Advanced Books & Software.

Thomée, V. (1969). Stability theory for partial difference operators. *SIAM Rev.*, **11**, 152–195.

Thompson, J. F., Warsi, Z. U. A., Mastin, C. W. (1985). *Numerical Grid Generation. Foundations and Applications*, New York, North-Holland.

Vorozhtsov, E. V. (1995). *Computer Algebra in Science and Engineering*, Symbolic-numeric stability analysis of difference schemes for compressible 3D Navier–Stokes equations, Fleischer, J., Grabmeier, J., Hehl, F. W., Küchlin, W. eds., pp. 340–356. Singapore, World Scientific.

Vorozhtsov, E. V., Yanenko, N. N. (1990). *Methods for the Localization of Singularities in Numerical Solutions of Gas Dynamics Problems*, New York, Berlin, Heidelberg, Springer-Verlag.

Warming, R. F., Beam, R. M., Hyett, B. J. (1975). Diagonalization and simultaneous symmetrization of the gas-dynamic matrices. *Math. Comput.*, **29**, 1037–1045.

Wolfram, S. (1991). *Mathematica: A System for Doing Mathematics by Computer*, Reading, Addison-Wesley.

Originally Received 15 April 1997
Accepted 04 December 1998